April 2011

Campustream: Mobile Social Networking

Bryan Richard Crabtree
Worcester Polytechnic Institute

Ryan Kenneth LeFevre
Worcester Polytechnic Institute

Campustream: Mobile Social Networking

A Major Qualifying Project Report
submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

_____

Bryan Crabtree

_____

Ryan LeFevre

Date: April 28, 2011

Approved:

1. Computers
2. Social Networking
3. Mobile Technology

_____

Professor Emmanuel O. Agu, Major Advisor

Abstract

The purpose of this Major Qualifying Project was to create a social network accessible from smartphones, which increased collaboration, and dissemination of information about campus events on a small college campus. Campustream consists of a website and an Android application that allows students to post interesting news stories, upcoming events, academic or social questions, and individual status updates. Posted material on Campustream can be voted on in order to determine an item's popularity. The web application is primarily implemented with PHP, and the Android application is implemented in Java. We performed a pre-survey to better understand the social networking needs of WPI students, and used this information while developing Campustream to make it custom tailored to WPI. In the end, we discovered some interesting social trends and some interesting opinions among the WPI student body. The current work done on Campustream also laid a stable foundation for future projects to build upon.

Acknowledgments

We would like to thank our advisor Professor Emmanuel Agu, of the WPI Computer Science department, for encouraging us, helping us, giving us creative freedom, and guiding us along the way. Both his interest and his dedication to Campustream have helped it grow to what it is today.

We would also like to acknowledge the numerous people who dedicate their time and effort to open-source projects. Especially, but not limited to, everyone who has contributed to: Ubuntu/Linux, Nginx, PHP, PHP-FPM, MySQL, Memcached, Redis, FlockDB, Android, and Reddit. Without them, Campustream would not have been feasible.

In addition, we would like to thank Emily Perlow, from the WPI Student Activities office, for allowing us to perform large scale testing among WPI students. Without her permission, we would have been unable to collect as much valuable data as we have been able to.

Finally, we would like to extend a big thanks to all of the WPI students that actively participated in our surveys and testing sessions. A social network is useless without users, and WPI students provided us with interesting data regarding social interaction on the Internet.

List of Tables

List of Figures

Table of Contents

1 Introduction

The introduction of the Internet just a few decades ago has brought about drastic changes in the way people live their everyday lives. Over the years the Internet has changed dramatically, going from a simple source of information, to a search-powered juggernaut, and from a single person experience, to one of the most socially interactive ways to meet and share experiences with others. In recent years, online social networks have started to thrive, and today the largest social networks host hundreds of millions of people, allowing new ideas that were never thought possible to come to fruition.

1.1 Project Goals

This MQP focused on social networking in the context of a college environment using smartphones. It brought together the social experiences of social networks, the mobility and advanced capabilities of today's modern smart phones, and the learning-based environment of colleges. The mobile nature of smartphones allows users to access social networks anywhere they desire and post about events while attending them. The main objective of this MQP was to test the viability of social networks in a college environment and to see what uses were facilitated by the introduction of a mobile platform. An overarching objective was to increase the attendance and awareness of social events on campus as well as to provide an easier way for students to interact and stay connected with one another.

1.2 Project Vision

By creating a complete social network that can be accessed either at home or anywhere on the Worcester Polytechnic Institute (WPI) campus, this project built a social platform that allowed WPI students to interact with each other in a more private environment that larger social networks do not provide. The vision of this project was to allow students to interact with each other and exchange information related to schoolwork as well as recreation. This project integrated two core social networking ideas:

1. Exchange of on-campus event information to allow students to see what events were happening on campus, when and where the event was located and see comments about an ongoing event to decide if it was worth attending.
2. Having multiple users all work together to solve academic problems. Students were able to ask questions and receive feedback from other students. These questions could have been

related to a specific class, a question about WPI in general, or just a general question about what was happening around the campus.

By integrating these two core social networking ideas, this project intended to keep students involved and interested in the mobile social network and allowed new interactions for all students involved. By limiting the social networks membership and use to the students and selected employees of WPI it also allowed a closer interaction between users without limiting their discussions to the types found on the existing MyWPI message boards. MyWPI is a web-based message board system, which allows students to discuss questions they have about a class, but does not allow them to ask questions about events or even classes they intend to take in the future. MyWPI message boards were typically set up and administered by professors for specific classes or projects. This MQP aimed to solve the problems illustrated by the following two usage scenarios, as well as many others:

1.2.1 Usage Scenario 1:

One common problem with college is that it can be difficult to keep up to date with social and academic events, as well as interesting news around the campus. Frequently, events and news postings are announced via email. While emails can get the attention of a wide audience, students have very little feedback on how the event turned out or what the event or news story is about beyond what the email tells them. This causes a distinct lack of social interaction around a socially centered event. Event organizers never get feedback on what events students enjoy or how to make events more interesting for students in the future.

By making event and news submissions student driven and student managed, a much more social aspect comes into play. If students can vote on what they feel is the most interesting news and events, then it becomes clear what the most interesting and important events and news stories really are. Uninteresting events are suppressed, while interesting events are actively propagated.

1.2.2 Usage Scenario 2:

Often students try to schedule classes so that they are able to take the class with a friend. Unfortunately, this doesn't always happen, and when a difficult homework assignment is given, they don't know who to talk to. There are a few solutions to this problem, but none seem to address the problem ideally. On one end of the communication spectrum, there is the class mailing list. Its main strong point is that it encourages discussion because everyone in the class receives messages sent to it. It's disadvantage, however, is that many students find mailing lists akin to spam because

of the large number of emails that can potentially be delivered with no way of unsubscribing. In the end, students ignore many emails including useful ones. On the other end, there is the Blackboard software (such as MyWPI), which features a discussion board. While this keeps large amounts of email out of student inboxes, it seems to reduce discussions and conversations. From experience, most students do not check the website to see if questions have been posted, let alone bother answering any questions.

The issues illustrated by the two scenarios above were the inspiration behind the creation of a Collaboration section of Campustream. We built a centralized place that fosters discussion without making the user feel like they are being spammed. Users were able to search for specific topics that they were interested in. Since our application had a friend graph, students were able to view only questions and answers posted by friends as an option.

2 Literature Review

2.1 Social Networking

Social networking on the Internet today is a ubiquitous part of our daily lives and is now one of the top ways to communicate with other people. It represents a revolutionary shift in communication. In fact, email, the de facto method of communication for many years on the internet, has recently been surpassed by social networking (The Nielsen Company, 2009), mostly due to Facebook's enormous growth.

2.1.1 History and Growth

The origins of social networking began soon after the creation of the public internet when numerous bulletin board systems (BBS) began to emerge. These allowed people to communicate with each other online and transfer files. While basic and independently coded, BBS's continued to rise in popularity all the way through the 1980's.

Perhaps the largest and earliest contributor to the world of social networking, was America Online (AOL). AOL helped expand the scope of the Internet from mostly hobbyists to the masses, and it featured user profiles and user communities (18).

Several years later, MySpace launched in 2003 and grew quickly to become the top social networking site on the internet, until the rise of Facebook, which has become the second most visited site on the internet.

Facebook was launched in 2004 and has experienced what is possibly the fastest growth of any website. In less than one year, between the launch of Facebook in February 2004 and December 2004, Facebook reached almost 1 million active users. Two years later in December 2006, Facebook had grown to 12 million active users. Just over three years later, in February of 2010, Facebook reached 400 million active users. It then took only 5 more months for Facebook to add 100 million new active users to reach an astounding 500 million active users on the site. In total, in 2010, Facebook added on average 20 million new users every month (6).

More recently, in 2006, the social networking site Twitter was founded. Twitter is a unique approach both because of its simplicity, and because of its ability to allow anyone to reach a large audience. It is a social network that is based on short status updates limited to 140 characters. Anyone can "follow" any other user on the network, which means they see all status updates from that user in real-time. Twitter's initial growth began during the 2007 South by Southwest festival

when they posted related tweets in real-time on large screens at the festival. It became such a hit that its daily post amount rose from 20,000 tweets per day to 60,000 (5). Four years later in June of 2010, Twitter had grown to 65 million tweets per day. As of this writing, Twitter has experienced a record of 3,065 tweets per second, which was due to the 2010 NBA Finals (16). There is no doubt that this record will be surpassed in the near future as Twitter continues to grow.

2.1.2 Privacy and Trust

The rise of social networking, unfortunately, did not come without a catch. Due to the friendly and social nature of social networking, people feel more comfortable posting personally sensitive information that they may not have felt comfortable revealing to the public in the past. Revealing such information allows criminals like sexual predators and stalkers easy access to information, although this is usually not a problem for most people.

According to *Privacy Protection Issues in Social Networking Sites* (13), privacy risks can be broken up into three categories: "Security risks (identity theft, phishing…), Reputation and Credibility risks (for example, companies do background checks on prospective employees…) and Profiling risks (spam, unsolicited collection of user data…)". Really, the main problems are that social networking sites do not inform or remind users of privacy issues, privacy tools are either overly complicated or too simple, and users cannot control what others say about them.

The paper proposes a tailored privacy framework that classifies different types of users and different types of privacy threats based on their social networking activity, as shown in the table below:

	Alpha Socializers	**Attention Seekers**	**Followers**	**Faithfuls**	**Functionals**
Numbers of Friends	Many	Many	Medium	Medium	Several
Principal types of Friends	Casual Friends	Casual Friends	Normal Friends	Normal Friends	Casual Friends
Spending times	Usually	Nearly always	Often	Less than often	Occasionally
Data	Lots of Photos, Comments, Tags, Activity	Lots of Photos, Comments, Blog, Tags, Activity	Some photos, Comments, Activity	Some photos, Comments, Activity	n/a
Data type	Mostly Harmful	Mostly Harmful, Poisonous	Harmless	Harmless	Harmless
Privacy Risks	Security, Reputation and Credibility	Security Reputation, and Credibility	Reputation and Credibility Profiling	Profiling	Profiling
Proposed Privacy Level	**Soft Privacy No Tracking**	**Soft Privacy No Tracking**	**Hard Privacy No Tracking**	**Soft Privacy No Tracking**	**Hard Privacy Soft Tracking**

Table 1: Privacy and Trust Framework

More recently, with the creation of "Geosocial Networking," or location-aware social networking, privacy has become more important than ever. Users still feel the comfort and security they felt before in social networks, and thus have no issues with broadcasting their exact location publically to anyone. Armed with such information, stalkers can track users down and perpetuate crimes. Applications such as Foursquare, Gowalla, Brightkite, SCVNGR, and now even Facebook, offer location-based services that allow users to share your location with both friends and the general public.

2.2 Social Network Questions

More and more people are turning to social networks to ask questions rather than using them to simply update their current status. Most people believe that social networks are helpful in finding practical information and for finding solutions to problems, ranging from simple to complex. According to a study centered around questions asked on social networks, most questions asked were short and to the point with an average length of only 75 characters, as opposed to lengthy explanations. In fact, the majority of questions asked are only a single sentence long (17). Part of

this is due to character limits imposed by the service, but even on services without this limit questions are often very short.

Interestingly enough, the most popular types of questions asked are looking for either recommendations or opinions. Most people have the expectation that questions broadcasted to a social network will have either biased opinions, or incomplete knowledge of the subject. In the case of friend-based social networks, such as Facebook, some people preferred asking questions over social networks because they felt they could trust their friends to provide more relevant answers than a search engine. The other reason people preferred asking a question over a social network is because they were looking for an opinion, not factual data, which they felt a search engine couldn't provide.

When users were asked why they answered questions posted in social networks, the top reasons were altruism, expertise, and interest in the topic (17). Most people feel good when they help someone else accomplish something, and social networks provide endless opportunities to do so with the least amount of effort required.

2.3 Case Study: Digg

Digg is a popular website whose primary focus is the sharing of links posted by users from around the Internet. These user-submitted links can be voted on, or "dugg," by users who find the website content interesting, thus generating a website that always shows the most popular current news stories. Digg is also very similar to another popular website called Reddit, which provided a lot of inspiration for parts of Campustream. By analyzing early comments and user activity on submitted stories, it's possible to predict the future popularity of an article. This fact was important because it meant that social behavior can, in some cases, be predictable.

The report *Digging Digg* has analyzed the various categories of Digg to generate numerous "egonets" which can be seen in Figure 1. These egonets organize users based on their most active category and shows the interactions between users. Green lines represent co-participation between users with the same category membership, whereas black lines represent co-participation between users of different categories [15].

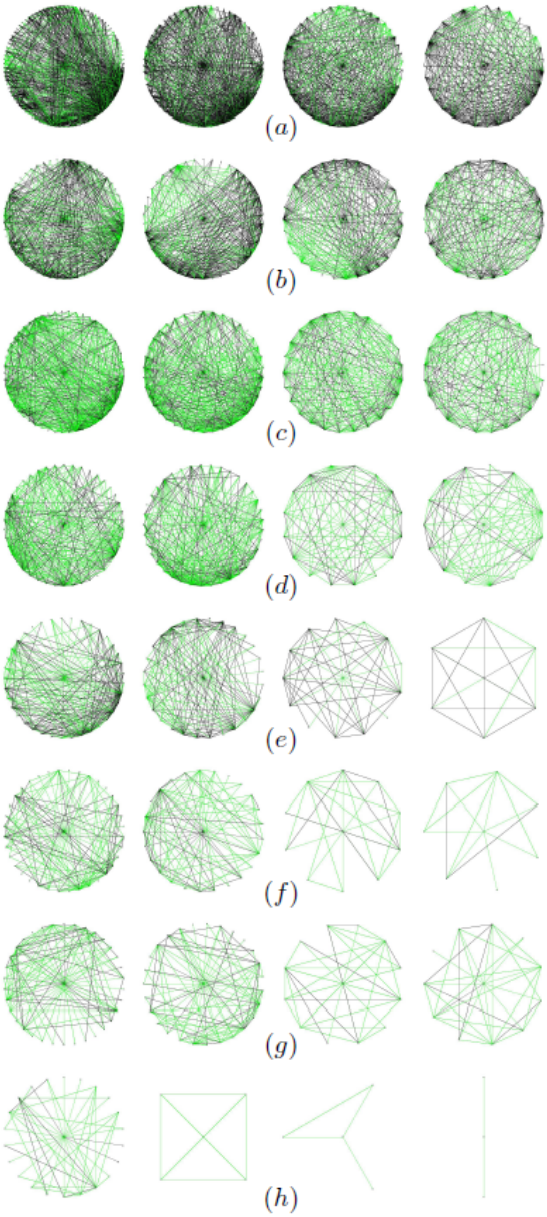


Figure 1: Egonets

Egonet graphs show that participation among certain groups, in this case World Business, is significantly higher than other groups. By analyzing data like this, one gets an initial feeling for the projected popularity of a certain piece of content. It seems plausible that applying this same type of analysis to other social networks, such as Twitter, can be used to predict the popularity of a topic in its early stages of growth. On Twitter, popular topics are called "Trending Topics," and are comprised of the most popular current topics being discussed on the social networking site.

The analysis done by the *Digging Digg* authors includes checking various things such as comment statistics, user interest peak, user feedback, and user community structure and membership. Combining all these various elements of a social network leads to a relatively accurate method of guessing content popularity. It also allows easy analysis of connections and identification of cases of isolation. Isolation specifically refers to user connections that are mostly separated from the rest of the social graph.

Isolation results when a few users only connect with each other and no one else in the graph. This means that either the users do not feel the need to be socially engaged with other users on the site, or there is a fundamental design flaw in the application. Analyzing social graphs in this manner has indicated to us the real intention behind the action of one user following another in a small network like Campustream, which we discuss in our results.

2.4 Web Application Technologies

The web portion of our application was built using many different web technologies. The primary language was PHP, and it used both a relational database and NoSQL databases.

Specifically, MySQL was the primary data store while Memcached[1] and Redis[2] provided a caching layer to speed up the application and provided more advanced functionality such as social streams. Campustream was optimized for high scalability, which unfortunately added to the complexity of the system, but certainly paid off in the end.

2.4.1 MySQL

MySQL is a very popular relational database used on a large number of websites, including some of the most visited websites on the Internet such as Facebook, Flickr, Wikipedia, and Youtube [21]. Because of this, it is known to be extremely stable and reliable as a data store. With the emergence of high-demand social applications however, its performance and scalability is unable to keep up without large server clusters.

High-traffic websites always implement a caching layer on top of MySQL that intercepts queries and returns the query results from data stored directly in memory as opposed to wasting precious resources searching the relational database. While MySQL does have built-in caching, it is very inefficient and not sufficient for high-performance applications.

MySQL allows for a cluster setup that usually involves a master-slave configuration where SELECT queries are run on a random slave server while all data-changing queries such as UPDATE, INSERT and DELETE are run on the master database and the change is replicated on all of the slave databases. It is vital that there is no lag between the master and slave databases in high-performance applications in this setup, or else the database queries will return inconsistent results depending on which server runs the query and how far behind the master database it is.

For our implementation, especially during the testing phase, we only used a single MySQL database because it had no problem handling the amount of load we initially experienced during initial development and testing. Using a single database initially simplified development and server administration.

2.4.2 Memcached

Memcached is also a very popular piece of software used by most high-traffic websites on the Internet. It's a fast, distributed, and in-memory data store that is often used as a database-caching layer. Facebook is the largest user of Memcached and has contributed many patches to its open source code-base in order to make it perform better across thousands of servers by reducing

[1] http://memcached.org/
[2] http://redis.io

its memory usage and improving how many simultaneous connections it can handle. While these improvements are more noticeable on massive implementations such as Facebook, smaller sites still benefit from the changes.

Memcached is a key/value data store, meaning that all data is stored and retrieved using a unique key that is passed through its API to identify the data. This unique key is a string that can range from a single word to an md5 hash depending on what is more convenient for the application developer. There are three main reasons why Memcached is so much faster than MySQL:

1. All Memcached data is fully stored in memory. This means Memcached avoids the slowness of disk lookup times by directly accessing memory, which also has a faster data transfer rate than disk drives.
2. Since Memcached is a key/value data store, it is basically equivalent to a hash table. This means it can retrieve data in O(1), or constant, time by jumping directly to the memory location that corresponds to the key given.
3. When using Memcached in a distributed environment, data is spread out over multiple servers using a global hash table, instead of a master-slave configuration like MySQL. This means that even data-altering calls only have to be run on the node where the data exists instead of on a single master node that would have to replicate the change to every node. This greatly reduces the amount of memory each node has to use and also improves the performance of Memcached significantly.

The framework we used for our application, named Hub, has a built-in caching solution called ActiveCache that closely models Ruby on Rail's ActiveRecord, but is written in PHP. ActiveCache provides an efficient and simple way to cache MySQL queries and will be explained in detail later in this paper.

Because Memcached data lives solely in memory, it is a *volatile* data store. This means that whenever the Memcached server or the whole computer is restarted all stored data is lost. This underscores the fact that Memcached should be used solely for caching data and should not be used to permanently store important information.

Memcached has the useful ability to set expire times for all pieces of data, which helps with memory management. If the server is running out of memory, Memcached will delete the least used data in order to make room for new data. It will also delete or overwrite any data that has passed its expiry time.

2.4.3 Redis

Redis is similar to Memcached in many ways, but it does have some very important differences that justify the use of both in a production environment. Although both Memcached and Redis access data fully from memory, Memcached is a *volatile* data store while Redis is a *persistent* data store. Redis achieves this by asynchronously saving data to the disk either when a certain period of time has passed, or a certain number of data-modifying queries have been made. Because this data saving call is asynchronous and periodic, it is not guaranteed to save 100% of data in the event that either Redis or the whole server crashes or restarts unexpectedly.

An important feature of Redis is that it has the *list* as a built-in data-type. This allows for some very useful applications such as creating in-memory queues with both head and tail access for O(1) constant access time to either end. The list *pop* and *push* operations used to take data off the front or back of the list is also atomic which means, even when multiple servers may be modifying the list, that none of the list data can be accessed simultaneously by two separate queries. These features are fundamental to high-performance applications where the probability of these events happening is rather common. The list data-type was crucial in storing data for our social streams.

Redis also offers the ability to act as a distributed data store like Memcached, but it is currently nowhere near as efficient as the Memcached implementation. This is why Memcached is better suited towards caching MySQL queries than Redis.

2.4.4 FlockDB

FlockDB is an open-source application written in Scala and developed by Twitter. It is the backend to Twitter's social graph, and it was used in our application for the same purpose.

FlockDB's main data store is MySQL, primarily because it provides reliability and ensures data integrity, but it also does some caching in the application itself. It uses sharding to break up the social graph into many pieces for faster access due to less data seeking in MySQL. It stores both forward and backwards paths between graph nodes in order to be as robust and efficient as possible.

2.4.5 Nginx

While the most popular web server available today is arguably Apache HTTP Server, it is not the most efficient. Nginx (pronounced "Engine X") is a very fast and scalable, web server that acts as an event-driven reverse proxy. Because it is multi-threaded and event-driven, it has a low memory footprint and can handle many times more concurrent connections than Apache can. It is

also easy to setup Nginx as a load-balancer to spread out high amounts of traffic among multiple Nginx web servers.

2.5 Development Platforms

A multitude of development options are available for modern smart phones. Two of the most commonly used development methods are to use either the operating system specific Software Development Kit (SDK), or to create a web based application. An SDK allows developers to access all functions of a specific platform or language. Both methods have their pros and cons. A specific SDK can reach down and use all of the core functions of the device, while a web application gives more limited access to what the device has to offer. On the other hand, a web application can be created to work on any mobile device with a browser, while a specific SDK only works on devices with that operating system. For example, using the Android SDK provided by Google allows the developer to use functions specific to Android phones so any code written will only work on Android devices. To determine which method will work best with this project's goals in mind, we considered three main points:

1. What percentage of the target audience can be reached with each development method?
2. What learning curve will be associated with each development method?
3. What desired features are we unable to implement with each development method?

By examining these three main areas we determined which development method to pursue, and which ones to forego.

2.5.1 Market Share

When examining the market share of each mobile operating system, we must consider several areas: The current market share of each operating system, the current rate of growth for each operating system, and the operating system's core demographic. While there are dozens of mobile operating systems available, three of those make up nearly 80% of smart phones that are currently in use. These are BlackBerry, iOS, and Android with 35%, 28%, and 15% of total market shares respectively [10]. These numbers alone don't tell the whole story, and we must look at many more aspects of these operating systems to determine how greatly each will affect our end goal.

Another major statistic to consider is the current growth trends for each of these operating systems. Over the past six months the numbers look slightly different, with BlackBerry, Android, and iOS taking 33%, 27%, and 23% shares respectively [10]. It is also important to note that

Android is the only mobile operating system that is actually increasing its overall market share at this time, with all others losing overall market share as each quarter passes.

The demographics of each smart phone are much more difficult to see with raw numbers. A few statistics can give us a general idea of how each mobile user is actually using their phone. One statistic is the percentage of web traffic each of these operating systems generates. As of September 2010, iOS, Android, and BlackBerry accounted for 56%, 25%, and 9% respectively [14]. Once again, Android is the only mobile operating system showing growth in this area, with all others losing market share as time passes. Another statistic we looked at is the number of active users that Facebook, the largest social networking website, has for each of its mobile applications. As of September 2010, Facebook had over 45 million active iOS users, 21 million active BlackBerry users, and over 3 million active Android users [7 8 9]. These numbers don't include mobile web site visits, or visits from unofficial Facebook applications.

All of these statistics point to one major conclusion, all three of these mobile operating systems have a large share of active users, and all three have a large number of users interested in social networking. While this project focused primarily on the use of Android and its capabilities with social networking on campus, the overall penetration of any social networking application will be severely limited by targeting a specific platform.

2.5.2 Development Options

There were three primary options for developing an Android application: The Android SDK, Google App Inventor, and a web application. The Android SDK is the most commonly used method for developing Android applications and has been used to produce over 70000 Android apps by developers around the world. Google App Inventor is a fairly new creation, and allows developers to drag and drop pieces of code and blocks of the interface to easily create complex applications. The final option is to make a web application; this allows the use of any existing web frameworks and will work with all mobile operating systems with a fully functioning browser.

The main advantages of using the Android SDK are the fact that it allows access to every feature Android has to offer, the SDK is heavily documented and fully supported by Google, allows applications to be placed in the Android Market, and is based on Java. The SDK also has many developer friendly features including an Eclipse plugin, an emulator to make testing easier, a suite of debugging tools, and much more [2].

The Android SDK does have a few disadvantages. First while Android is based on Java it doesn't use the standard Java libraries. This means coding can be done in the Java language, but access to many of the advanced Java packages is not available [11]. In some cases this can mean reinventing the wheel to do something Java has allowed for many years. Another disadvantage is that Android does not offer a WYSIWYG system for developing application interfaces. There are several third party tools available such as Droid Draw, however these tools are not directly supported and do not offer functionality for all types of Android interfaces [3]. The last main disadvantage to programming with the Android SDK is that applications cannot easily be ported over to other devices. iOS does not allow Java programming and BlackBerrys are similar to Android in that they use Java, however they use their own library of Java packages and not the standard Java packages.

A newer option available to Android developers is the use of Google App Inventor. This tool is based on MIT's Open Java Blocks Library and allows the developer to simply drag and drop sections of code and edit the exact properties of what each block does. This tool allows access to many but not all of the features of Android phones, these include GPS information, accelerometer data, speech-to-text capabilities, and persistent storage on the web [4]. These are many of the features we had planned to use for this project, and in a sense of accessible features, App Inventor offers most of what we needed. However one major issue is that developers are forced to use already created blocks of code, and cannot add their own. Besides this issue, App Inventor's other main problem is that it limits the application created to the Android platform, without the possibility of being ported to another platform. It also features a useful WYSIWYG editor that is much faster than trial-and-error design methods.

The final development option to consider is creating a web application. A web application allows the use of open web standards that are viewable in any browser. The obvious advantages to this method are that the application will be available to any smart phone that has a capable browser. This will also allow the application to be viewed on a computer, which some users may prefer to do.

The main drawback to making a web application is that it would have severely limited what features of the phone we had access to as developers. While web applications can now access GPS data, it is less accurate and takes longer than methods built into the phone's native APIs [19]. The most common system for determining GPS coordinates through the web is the Geolocation API Specification. This system has a high accuracy mode, however this mode takes a long time to activate and requires additional resources, severely limiting how well a GPS could be used [19]. Accelerometer and orientation data are entirely unavailable and there is no access to many of Google's APIs.

	Android SDK	Google App Inventor	Web Application
Language	Java Based	Blocks Based	Any
Development Time	Standard	Short	Standard
Documentation	Heavy	Some	Heavy
Audience	Android	Android	Everyone
Market Access	Yes	No	No
GPS Access	Yes	Yes	Partial
Accelerometer Access	Yes	Yes	No
Calendar Access	Yes	Yes	No
Messaging Access	Yes	Yes	No
API Access	Full	Partial	None
GUI Builder	No	Yes	Yes

Table 2: Android Development Options

All three development methods have obvious drawbacks and advantages. The above table offers a comparison of the three development methods and summarizes the reasons for using each. We weighed the pros and cons of each method and determined how each method was limited not just in general, but in a campus environment. We determined that some of the drawbacks of a particular method were nullified when we compared it to the demographic and environment of the WPI campus.

3 Pre-Development Survey

Before embarking on the design and implementation of Campustream, our mobile social networking application, we conducted a survey among WPI undergraduates comprised of various questions related to social networking and our project's goals. Our goal was to better understand the social networking needs of WPI students. We received 534 responses, and below is a full analysis of the data gathered and how it affected our project. At the time of the survey, WPI has 3,453 undergraduates with 28% women and 72% men attending from 36 different countries around the world[3].

The survey was conducted using Google Docs Forms, which also provided the response graphs. Data was collected over a period of 3 days from October 3rd until October 6th, 2010. This was done by sending an email to the undergraduate list at WPI, which reached over 3000 students. This email was sent in the evening on Sunday, October 3rd because we believed students would be most likely to respond at that time as they would be doing other assignments for the week. A copy of the survey sent to students can be found in Appendix E.

3.1 Questions and Responses

Which of the following mobile devices do you own?

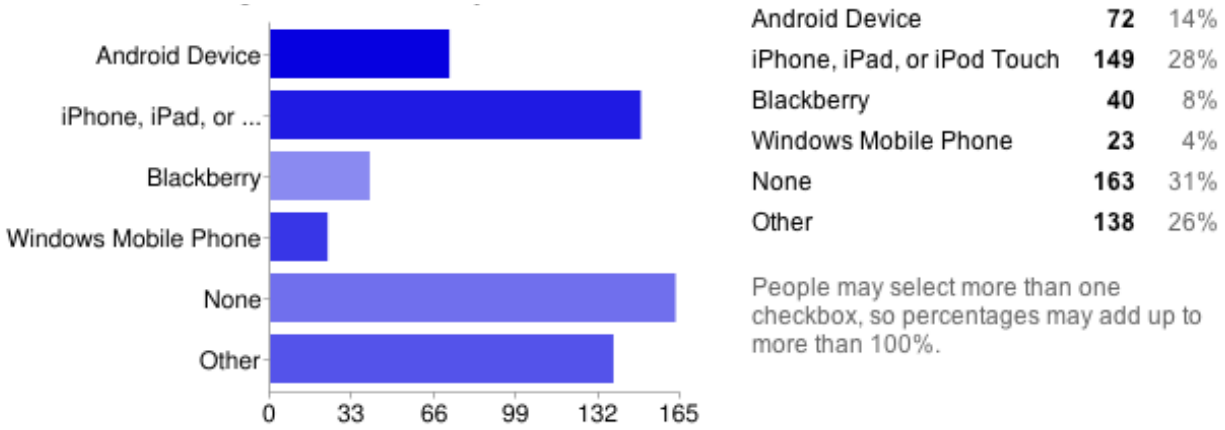


Figure 2: Breakdown of Mobile Devices among Students

This question helped give us a feel for the distribution of smartphone ownership among WPI students, who are our primary testing group.

[3] http://www.wpi.edu/about/facts.html

As shown by the data, the ownership of mobile devices is quite varied. The majority of WPI students do not own a smartphone; especially considering most of the 'Other' category can be merged with the 'None' category (most responses that fell under 'Other' occurred when students typed in their model of non-smartphone).

Among the smartphone users however, the iOS devices were the clear winner, with about 28% of WPI students owning at least one. Behind the iOS is the Android OS, with about 14% of WPI students claiming ownership. Blackberry and Windows Mobile Phone ownership is very low, most likely because their products are targeted more towards corporate business users while iOS and Android are more generally used.

Because there is so much fragmentation between smartphone platforms, the creation of a web application was vital. In addition to creating a web application, we also created a native Android application. Deploying the Android application in conjunction with a web application helped expand the reach of the network.

Do you intend to purchase any of the following mobile devices in the next six months?

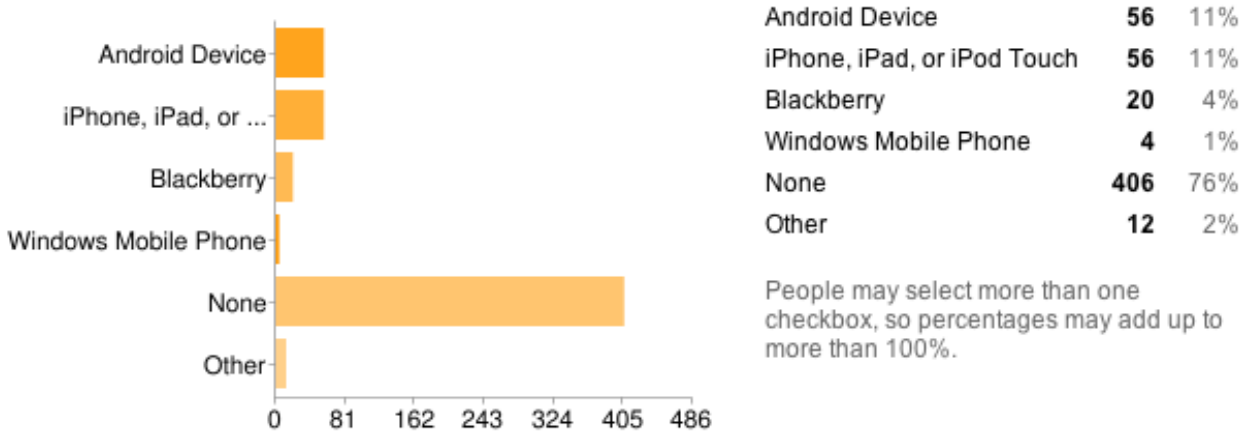


Figure 3: Intent of Students to Purchase a Mobile Device in the Future

For this question, we wanted to predict what the future trend for the mobile smartphone market is among WPI students.

Perhaps the most interesting point here is that students were just as interested in buying an Android phone as they are an iOS device, although both of these percentages are only 11% each. Overall, there doesn't seem to be a very strong smartphone adoption rate among college students at this time.

Which of the following social networks do you currently participate in?

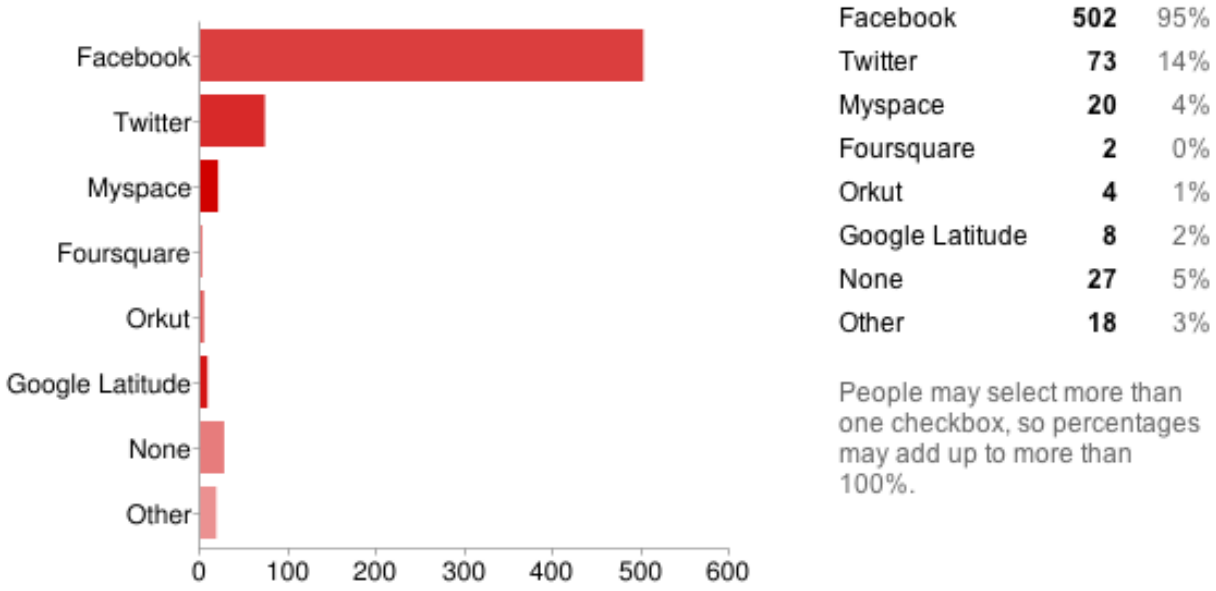


Figure 4: Student Social Network Participation

We asked this question because we wanted to get a feel for what kind of social networking experience students at WPI are used to. According to the survey, 95% of WPI students used Facebook while only 14% used Twitter. While it is not impossible that Facebook could have a higher percentage than Twitter, it should be noted that at the time of the survey Facebook had over 500 million users while Twitter had fewer than 200 million.

Beyond Facebook and Twitter, there really seems to be very little social network usage. Only 2 students used Foursquare, which is less than 1% of students who took the survey. Some examples of social networks provided with the 'Other' option include: LinkedIn (a business oriented social network), Ping (Apple's music social network), and Hi5 (an entertainment-based social network). Only 3% of students used these other social networks in total.

The last important thing to note is that only 5% of students said they do not participate in any social networks. This means almost all of WPI's students are familiar with how social networking works, how to participate, and how to leverage their services.

Do you actively use location-based services in social networks?

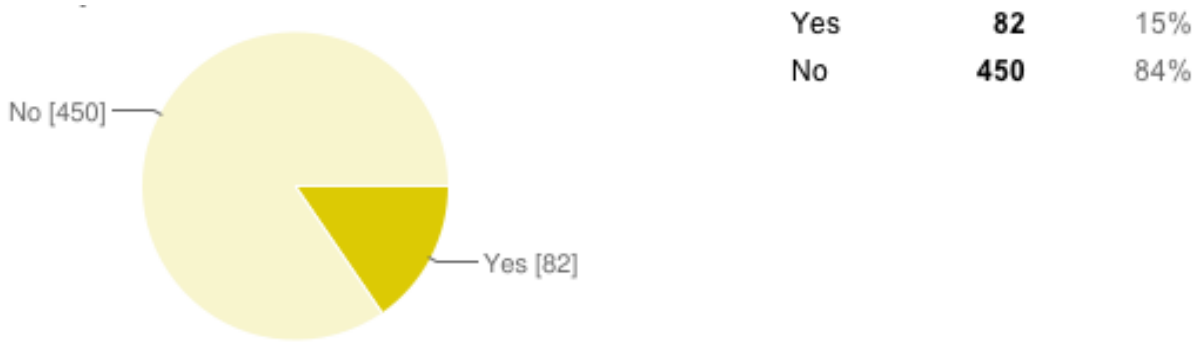
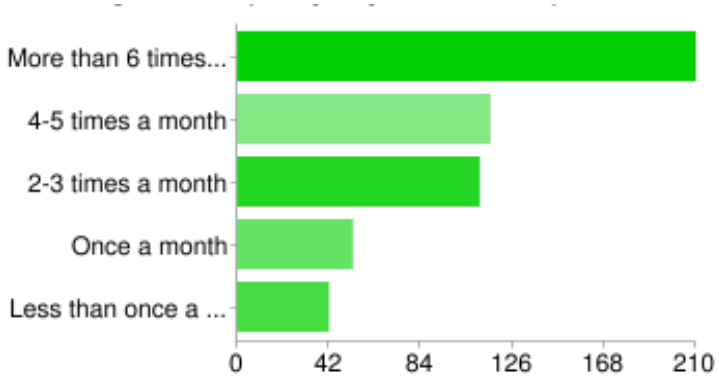


Figure 5: Student Location-Based Service Usage

The primary focus of this question was to determine how comfortable students were with sharing their current location, whether it is privately or publically.

Only 15% of students said they share their location online. Since only 2 students said they use Foursquare, it is presumed that the rest of the students use the newly launched Facebook Places to share their location. The other possibility is that the question was too vague and students thought the question also included things such as using GPS map navigation for getting directions from your current location. The question was clarified with "In other words, do you use services that focus on sharing information related to your current location?", but this may have been easily overlooked. Overall, this particular piece of data should be interpreted with that problem under consideration. Initially, we had considered adding location-awareness to Campustream, but since relatively few students shared location information online, we decided to spend our time developing other social networking features that students were more interested in.

On average, how frequently do you attend campus events and/or social gatherings?



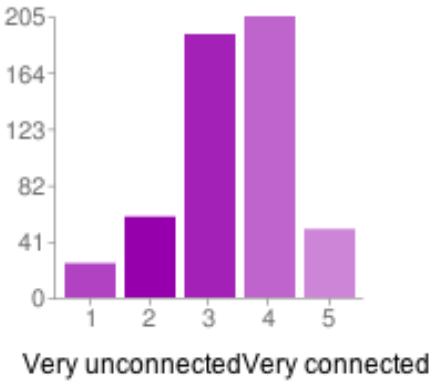
More than 6 times a month	**210**	39%
4-5 times a month	**116**	22%
2-3 times a month	**111**	21%
Once a month	**53**	10%
Less than once a month	**42**	8%

Figure 6: Student Social Event Attendance Frequency

In order to get a feel for how active WPI students are, we wanted to find out how often they attended events on campus. We clarified this question by saying: "Including school sponsored events, fraternity/sorority parties, club/organization events, etc."

Overall, WPI students are fairly socially active, with 61% of them attending four or more events per month. 39% said they attend more than 6 events per month. This result clearly supports the need for our application, because it means WPI students are interested in attending events which increases the chances that they would be interested in using our application.

On a scale of 1 to 5, how well connected do you feel with other students and campus events?



1 -Very unconnected	**25**	5%
2	**59**	11%
3	**192**	36%
4	**205**	38%
5 -Very connected	**50**	9%

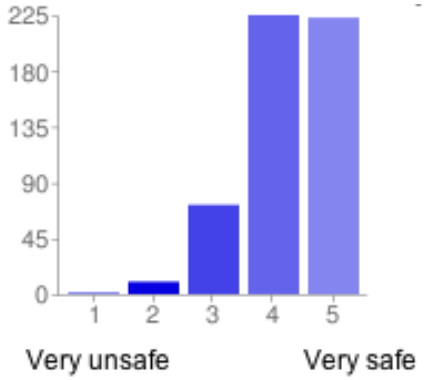
Figure 7: Students Level of Connectedness with other Students

In accordance with the previous question, we wanted to know if students felt like they were well informed about social events around campus. We clarified this question by adding: "Do you

feel like you always know what is going on around campus or do you feel like you're missing out on a lot?"

The vast majority of students reported that they fall around the category of "somewhat connected". This means that students usually know about some campus events, but they also feel like they are missing out on others because they are not well informed about them. This also means that WPI students only feel somewhat connected to other students. Our application aims to improve this metric, so it is good to know where our starting point is when we launch it for testing.

On a scale of 1 to 5, how safe do you feel on campus?



1 -	Very unsafe	**1**	0%
2		**10**	2%
3		**72**	13%
4		**225**	42%
5 -	Very safe	**223**	42%

Figure 8: Students Self-Described level of Safety on Campus

This question was directly related at the security portion of our planned application. We wanted to get a feel for the safety around campus, and therefore the amount of usage this portion of the application would get.

The results clearly show that most people think WPI is a very secure campus, and that most people don't feel threatened walking around campus, even at night. However, the responses to a later question asking if students would be interested in using a social application for monitoring suspicious activity seem to show quite the opposite.

Would you be interested in using a mobile application that aims to improve social connections on campus and provide a central place to learn about events?

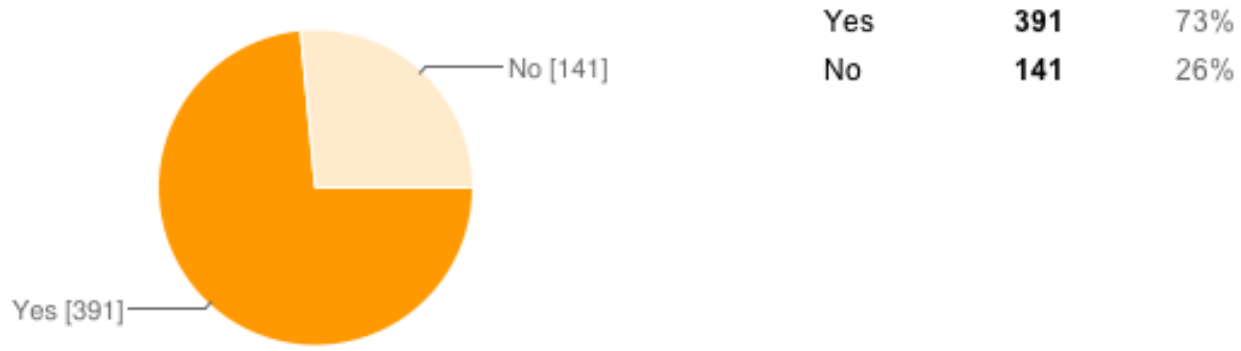


Figure 9: Interest in a Mobile Application that improves Social Connections

This question, along with the next few questions quantified how much of an interest there was for our application in general, as well as for various components of our application. For this question, we were interested in the events component and how many students would potentially use it.

It turns out that this component actually had the least amount of interest, while still maintaining a majority with 73% of students saying they would be interested in using this part of the application. Because of this, we designed the application bearing in mind that the events component may get less use than other components.

Would you be interested in a system that allows you to collaboratively ask questions among your peers?

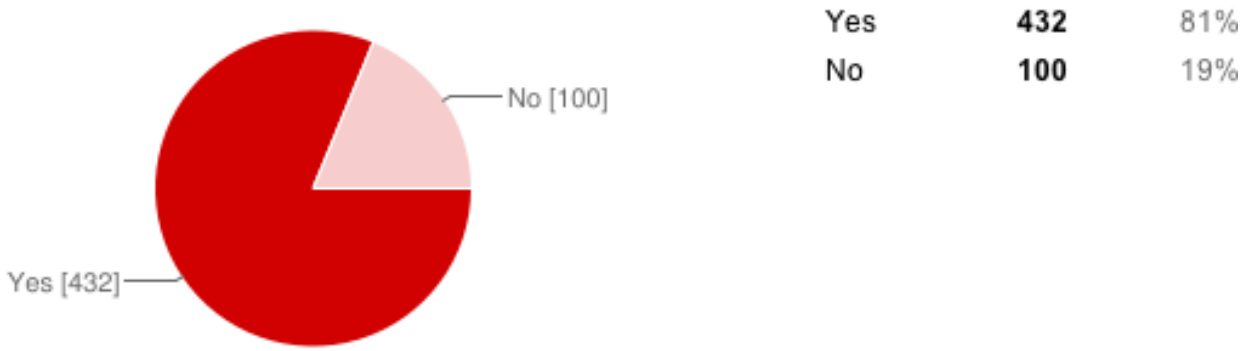


Figure 10: Interest in a Collaboration System

This question directly aimed at judging interest in the collaboration component of our application. We clarified the question with: "This includes class-related questions, social questions, and anonymous questions" in order to give the student some possible use cases.

The collaboration component of our application is arguably the most "social" component, and accordingly it seemed to have generated a lot of interest. 81% of students who took the survey expressed interest in using an application that provides this functionality.

Facebook provides a social question and answer service, but it is very broad. Facebook questions are shown to everyone on Facebook with similar interests and are not restricted to specific groups or specific locations. Our application was customized for WPI students and was exclusive to WPI so that students could ask very specific questions that people outside of WPI wouldn't find interesting or be able to answer.

Would you like to see a social application that helps report and monitor suspicious activity on campus?

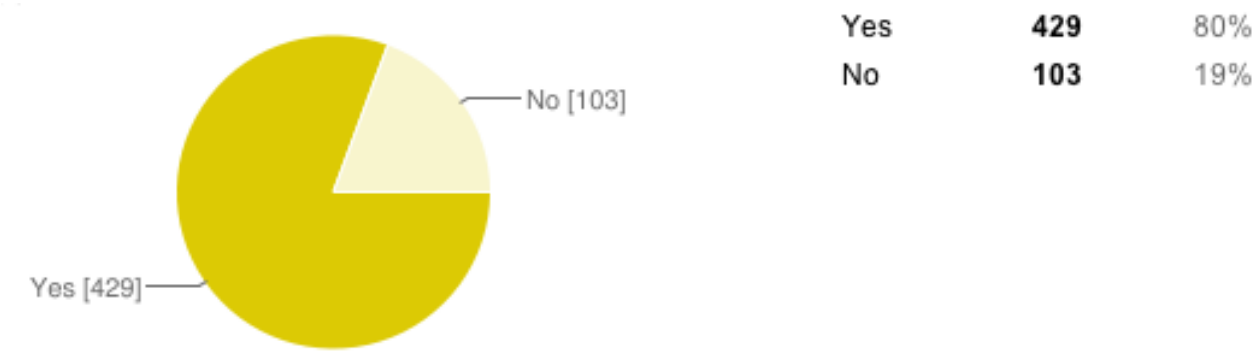


Figure 11: Interest in a Social and Mobile Security Application

As mentioned before, the results to this question seem to strangely clash with the results of the question asking if students felt safe on campus. While the vast majority of students said they did feel safe, 80% still said that they would be interested in using an application that helps report suspicious activity.

The combination of these two metrics could mean that, although students feel safe on campus, they feel that there is nothing outside of calling the police whenever they see possibly suspicious activity. It's possible that students may see suspicious activity, but would be hesitant to call the campus police as they do not want to waste the police force's time if it turns out to be a false

alarm. Our application could be used to discuss minor events that would otherwise not be reported and could also help in investigations when major security breaches did occur. In all cases, we did not advocate using our application in place of calling campus police. The police should definitely be notified for all major security problems.

Would you be willing to participate in testing a social networking application on campus?

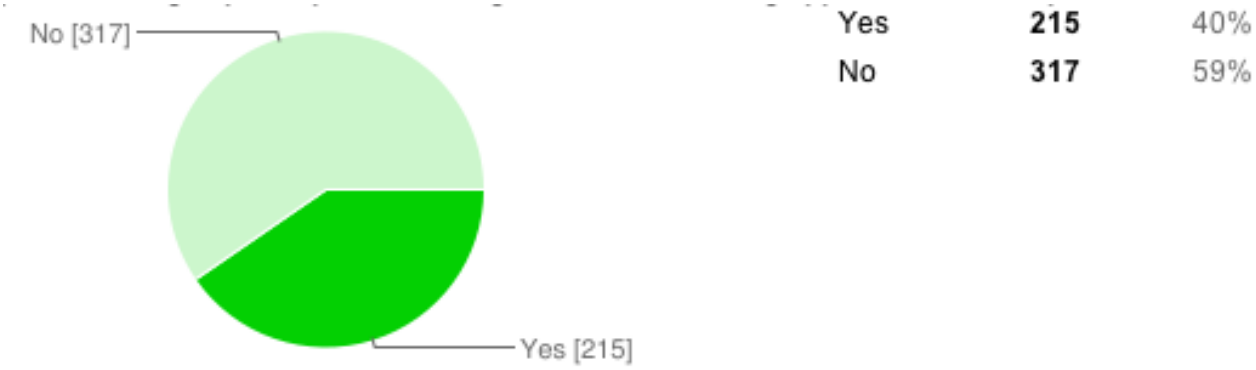


Figure 12: Interest in Participating in a Social Network-Testing Group

This final question was asked so that we would have a feel for how many students we would have available for beta testing when the time arrives. Students were asked for their email address if they expressed interest in testing our application so that we were able to get in touch with them when testing began. 215 or 40% of students said they were interested in testing the application, which implied we would have significant traffic and usage of our application.

4 System Architecture

Before design started on the project, we knew we wanted to build a highly scalable system that would be able to handle hundreds of requests per second. We also knew that we wanted to be able to efficiently store a large amount of data, and in order to do so, we would need to install many various open-source pieces of software such as Redis, Memcached, FlockDB, and so on. Shared hosting providers do not offer this ability, and dedicated servers can be very expensive. Thus, the logical choice would be a virtual private server (VPS) or cloud server, since it would give us full root access to the system at a discount price.

Cloud servers have been quickly rising in popularity, especially because of their low cost to both end-users and server hosts. Since a single physical server can host multiple cloud servers, hosting companies can charge significantly less, while providing the same flexibility as a dedicated server. Some hosting companies, such as Amazon and MediaTemple, even distribute cloud servers over multiple physical servers to add increased redundancy and reliability. Also, since cloud servers are simply virtualization instances, they can be easily cloned and copied to rapidly scale any hosting infrastructure.

Ideally, we wanted to host the site on-campus at WPI to avoid hosting costs, but we feared we would run into too many problems along the way. There are no servers set up on campus where we could openly begin hosting a site, so we would need to purchase our own hardware, which would be a costly investment. Secondly, WPI has strict bandwidth limitations, as enforced by Network Operations, and we did not want to worry about violating those restrictions.

4.1 Server Hosting

There are many different options to choose from regarding web hosting. Since our application used specific web technologies that normally are not offered with shared hosting plans, the next best options are either cloud hosting or a virtual private server. These cloud hosting options were significantly cheaper than renting an entire dedicated server, yet they provided us with full root access to the server and the illusion that we were using a dedicated server. Their resources were more than enough for our needs, especially during the testing and development phase of the application.

The table below shows a comparison of numerous web hosts we considered for hosting the web portion of our social network. The values listed are all taken from the cheapest plan the companies' offered since each plan offered enough resources for our needs in all cases.

Server Host	Server Type	Monthly Cost	Server RAM	Server Disk	Bandwidth In	Bandwidth Out
Rackspace Cloud	Cloud VPS	$10.95	256MB	10GB	22 ¢/GB	8 ¢/GB
Linode	VPS	$19.95	512MB	16GB	200GB max combined	200GB max combined
Amazon EC2 Micro Instance	Cloud VPS	$14.00	613MB	$0.10 per GB-month of storage	10 ¢/GB	15 ¢/GB
Site5 Virtual Servers	VPS	$50	756MB	40GB	600GB max combined	600GB max combined
Slicehost	VPS	$20	256MB	10GB	150GB max combined	150GB max combined

Table 3: Server Hosting Comparison

As shown by the data in Table 3, some web hosts provide unlimited bandwidth because they charge a certain amount per GB transferred, while others charge a flat rate and impose a bandwidth cap. In all of the latter cases, the bandwidth is calculated as a combination of both upload and download amounts. All of the VPS hosts use the fixed bandwidth cap system, while all the cloud hosting providers charge by the gigabyte. The only host that charges extra for disk space is Amazon EC2, which charges 10 cents for every gigabyte used per month. For our estimated usage, this charge was minimal, and also allowed for unlimited data storage capabilities.

Since the two cloud hosting plans that we investigated charged for bandwidth, the monthly cost given did not truly show the monthly expenses required for each plan. Since 10 gigabytes of space was adequate for our needs during testing, below is a breakdown of bandwidth costs for Rackspace Cloud hosting.

Bandwidth In (right) Bandwidth Out (below)	1GB	2GB	5GB	10GB
1GB	$0.30	$0.52	$1.18	$2.28
2GB	$0.38	$0.60	$1.26	$2.36
5GB	$0.62	$0.84	$1.50	$2.60
10GB	$1.02	$1.24	$1.90	$3.00

Table 4: Rackspace Cloud Bandwidth Charges

4.2 Social Network Aggregation

Since our pre-survey showed that over 95 percent of WPI students used Facebook, we felt it would be easier for students to sign up using their Facebook accounts and also to allow them to cross-post material on Campustream as well as Facebook. In order to make the platform more inviting and useful for its users, both Facebook and Twitter were integrated.

For Twitter, we also gave users the option to broadcast various bits of info from Campustream to their Twitter account including events, news postings, answered questions, and so on.

4.3 Social Streams

In order to make it easy for users to see the latest activity on the site from both the people they follow and all of the students who choose to broadcast their actions publically, multiple social streams had to be created on the site. The streams were perhaps the most important aspect of the website since they were the central place to see what everyone was talking about.

The friend stream showed updates only from people a user followed. These updates include status updates, comments posted, questions asked, and news and events created. The public stream showed updates from everyone who decided to share content publically, and provided a simple way for users to find people to follow and to see all the latest information on the site.

Each user also had an activity stream that showed on their profile page and was the main focus of each user's page, in addition to some interesting information about the user such as a short bio, their class year, and their majors.

4.4 Web Application Architecture

The web application architecture was split between two separate servers: a webserver and a database server. Both servers had distinct roles in the operation of Campustream. As shown in Figure 13, each server ran a combination of important services.

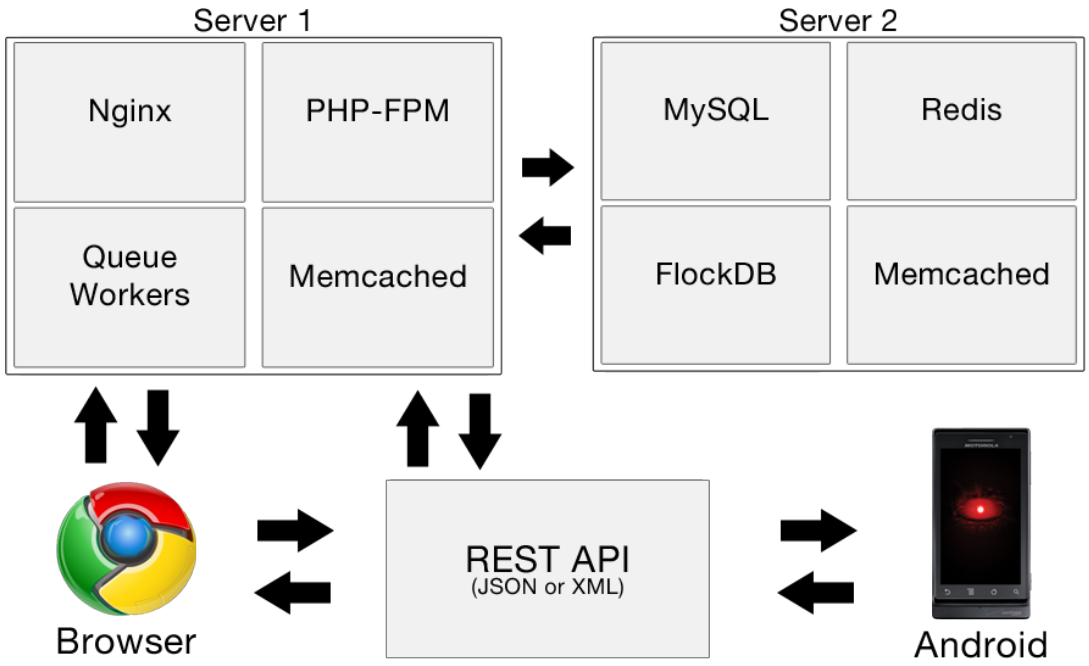


Figure 13: High-level System Architecture for Campustream

The web server handled all incoming web requests, processed them, and returned a result to the user. The application ran on this web server. The database server (server 2) ran all database-related systems. All the data for Campustream is stored on and retrieved from the database server.

By splitting up Campustream into two separate servers, the load on all systems is distributed. The load on the web server, handles user requests, no longer depends on database load. Since the servers used by Campustream have relatively small system resources, this also ensures that all systems have enough resources to run efficiently.

The API for Campustream is a simple Representational State Transfer (REST) interface that returns either JavaScript Object Notation (JSON) or Extensible Markup Language (XML). All the HTTP POST requests to the API require authentication, and all the HTTP GET requests do not. API calls that require authentication modify the system state in some way, while non-authenticated

calls simply retrieve unprotected data. A full reference for the Campustream API can be found in Appendix C.

4.4.1 Nginx

All requests to the web server were routed through Nginx, which acted as a reverse-proxy to PHP-FPM. In other words, Nginx acted as a proxy on the server itself by passing requests to PHP-FPM, and then returning data to the client as if the data came from the proxy itself. Once the application finished running, Nginx returned the response to the user. All requests to and responses from Nginx were done over HTTP. Since Nginx does not have a built-in PHP module, PHP-FPM is required because Campustream is written in PHP.

4.4.2 PHP and PHP-FPM

Campustream's web application was written in PHP. PHP-FPM managed the PHP processes running on the web server, and communicated with Nginx over FastCGI. FastCGI is a common protocol for interactive applications to communicate with web servers, and is an improved version of the old Common Gateway Interface (CGI).

When an incoming request from Nginx was encountered over FastCGI, PHP-FPM executed a PHP instance with that request, and returned the output to Nginx. During the life of the PHP process, many different services were used to complete the request.

4.4.3 Memcached

Memcached was used as an object-store, primarily for results returned from MySQL. It greatly sped up the execution of the web application and was critical to the application's overall speed. Memcached ran on both the web and database servers in order to provide as much memory for caching data as possible, and to split up the load between the two servers.

4.4.4 Queue Workers

In order to keep the web interface and Android application consistently fast, queue workers handled some potentially long running processes asynchronously. The queue workers consisted of cron jobs that handled things such as confirmation emails, notification emails, and status exporting to Twitter and Facebook.

4.4.5 MySQL

MySQL was the primary data store for Campustream. All the data collected from the site and entered by users was stored in MySQL, but also mirrored to some other services such as Memcached and Redis for speed purposes.

4.4.6 Redis

Redis stored all the site streams as lists, and also stored all the queues that the queue workers interacted with. In order to keep memory usage to a minimum, Redis only stored the MySQL primary IDs for each user status, instead of a serialized object or the status message itself.

4.4.7 FlockDB

FlockDB's single purpose in Campustream was to store the social graph. All social graph queries were executed directly by FlockDB and the results were returned to the PHP module running on the web server.

4.5 Android Application Design

All of the design decisions for the web portion of the project heavily correlate with their Android counterparts. The idea was to make an application that acted exactly as the website and featured all of the same functions. All of the same social streams and categories are present, users can find friends, check someone's profiles, and post new content to the network all from the Android application.

The application retrieves all of its information from the web API found in Appendix B. This requires the application to open an HTTP connection through the devices current Wi-Fi or mobile network connection. The device then makes a HTTP call to the API and uses the authentication the user provided at login to ensure security.

The application itself uses very few of Google's APIs and primarily uses code already available in Java. All of the HTTP connections and image downloading is done with standard Java libraries. External libraries include XML and Json parsers to parse the information retrieved from the API calls. All API calls return a Json or XML string which then needs to be parsed and the relevant information retrieved.

We originally planned to use the Google Maps API to allow users to see location information about events and also to enable mobile check-ins. However, these ideas were scrapped due to user interest and the time required for their implementation. All Facebook and Twitter connection

information is done on the website, so APIs for those services were unnecessary for the Android application.

5 Implementation

The implementation of the project was tightly integrated with the overall design of the project. Both the web and Android clients were based on the ideas discussed in the design section of this report. The Android client saw the most change from the original design due to the removal of many location-aware features and the natural look of the more recent Android versions.

5.1 Web Application Framework

In order to facilitate and speed up development, an application framework was used. As mentioned before, all of Campustream's web application is written in PHP. The PHP code used in Campustream was developed with PHP versions 5.3.x or later in mind due to the improved object-oriented design of these PHP versions and their support for anonymous functions.

5.1.1 Framework Overview

Campustream uses a Model-View-Controller (MVC) framework developed by TwitPic named Hub. At the time of this writing, this MVC framework is not open-sourced yet, but there are plans to release it as open-source in the future. Special permission was obtained from TwitPic prior to using it in any way.

Hub facilitates the rapid development of high-traffic websites. It was built with MySQL clusters and Memcached in mind, although it is flexible and both are optional.

In Hub, *Models* are PHP objects that are usually a direct mapping of database tables. It uses a system called ActiveRecord, which is a part of Hub and is inspired by Ruby's ActiveRecord class, to make database queries and return the results in the form of one or more Models. Hub also has ActiveCache, which is an extension of ActiveRecord and is used to cache database query results with Memcached automatically. An example of some data retrieval using ActiveCache would be:

```
$user = ActiveCache::find('User_Model', "user:$name", 43200)->sql(
      "SELECT * FROM users WHERE username = '$user' LIMIT 1"
);
```

This example checks Memcached to see if there is any data stored at the key "user:$name" (where $name is a variable representing a String). If the key exists in Memcached, it will immediately return the stored data. If the key is not in Memcached, it will query MySQL with the provided query, retrieve the data, store the data in Memcached for future retrievals, and then return the data.

Controllers are used to perform most of the application logic that is needed for the requested page. Most Controllers will load one or more Models, perform some additional logic if necessary, and then send the data to a *View.*

A *View* contains the HTML that ultimately gets sent to the browser after some processing. Normally it is a good goal to keep as much PHP logic out of the *View* as possible because it promotes cleaner HTML that is easier to work with and modify later if necessary. Unlike some other PHP frameworks, Hub does not use a templating system. Instead, it lets you access the contents of PHP variables directly, giving the developer much more flexibility.

Static files, such as JavaScript or Cascading Style Sheets (CSS), are served from the www directory. Nginx is configured such that URLs to files that exist are not rewritten using URL rewrite rules. This means that any file in the www directory that is directly requested will be directly served by Nginx and will bypass Hub altogether.

A full PHP code example of the User_Controller can be found in Appendix G.

5.1.2 Routing Requests

When Hub is invoked by PHP, it has to figure out what actions it needs to take in order to correctly process the request, which is handled by the router. The router is pre-configured with a set of *routes* through the application. These routes relate URLs to Controllers and *actions* (methods in the Controllers that handle requests).

Nginx performs URL rewriting in order to inform Hub of the visited URL. For example, if the user visits:

```
http://campustream.com/user/ryan
```

Then, the URL will be modified by regular expressions to become:

```
http://campustream.com/index.php?hub_uri=/user/ryan
```

Thus, all requests begin with the index.php file, and the route is specified via a HTTP GET variable. Hub inspects this GET variable and compares it against a list of routes from the application configuration. The routes can be defined as simple static routes such as “collaboration”, or they can be defined with regular expressions in order to support wildcard matching. In the case of user profiles, the route from the configuration looks like:

```
$config['user/([A-Za-z0-9_]+)'] = 'user/show/username:$1';
```

The index of the array is the regex that is compared against the hub_uri GET variable, and the corresponding array value is the controller and action, as well as the argument to pass to the action (username, in this case).

The router is also responsible for determining the response type of the request: whether the user wants JSON, XML or HTML.

5.1.3 Controller Actions

Once the correct route for the incoming request has been found, the corresponding controller is loaded and the specified action is executed. In the example from the previous section, the User_Controller is used to process the request, and the show() method is invoked.

Once the controller action is invoked, it is responsible for handling the lifetime of the request. It gathers all data required in order to process the request, and then instantiates one or more views to render the response data. Data is gathered through various methods such as ActiveRecord/ActiveCache, PhlockDB (PHP FlockDB library), Predis (PHP Redis library), or Memcached (PHP Memcached library).

For example, if the controller needs to load a user, it would execute:

```
$user = ActiveCache::find('User_Model', 'name:$username', 43200)->sql(
      "SELECT * FROM users WHERE username = '$username' LIMIT 1"
);
if (!$user->is_loaded()) {
      View::respond_to("html", function () {
            echo "Not found.";
      });
      View::respond_to(array("xml","json"), function () {
            Hub::http_error(404, "User not found");
      });
      return false;
}
```

The above code first attempts to load the user directly from Memcached. If the user is not cached, it then falls back to loading the user from MySQL. If the user is found, ActiveCache caches the user in

Memcached, then returns the User_Model object. The controller then verifies the model to make sure that it was found, and if it wasn't, invokes an error response and quits.

If the response type, as reported by the router, is HTML, then HTML to render the webpage is returned. If the response type is JSON or XML, then the data is printed in the appropriate format. When responding with JSON or XML, the controller can perform:

```
View::respond_to(array('xml', 'json'), function ($format) use($user) {
      echo $user->{"to_$format"}();
});
```

In this block of code, a few important things are happening. First, the respond_to() method is told that the callback in the second argument should be executed if the response type is XML or JSON. Since that is the case in this example, the callback is executed with the requested format as the argument. The User_Model ($user) is included in the callback's closure using the PHP use() function. Inside of the callback, the to_json() or to_xml() method is invoked, depending on the response type that was requested. ActiveRecord converts the User_Model to the corresponding type, and then the contents are simply printed. A similar block of code is used for HTML requests, except a view is rendered instead of a model, as shown in the next section.

5.1.4 Views

If the request response type is HTML, then a view will be instantiated to render the data (view objects are not needed if the response type is JSON or XML) and data is passed to the view through class variables. Once the view is given all the data it needs, it is rendered.

Controllers can also define templates, which are automatically instantiated views when the controller is created. These templates represent a base site layout that is used on every page, and then specific page views are injected into it. For example, the following code would be inside a View::respond_to() callback function:

```
$view = new View("user/show");
$view->user = $user;
$template->content = $view->render();
echo $template->render();
```

This is used in User_Controller->show() to show user profiles. The main site template is automatically loaded, and the profile page is first rendered, then injected into the template. Finally, the template itself is rendered, which now includes the profile HTML page. Inside the user/show

view, the page can access the user data set in the controller via the $user variable, since that is what we set before rendering the view.

5.1.5 REST API

Models in Hub are designed such that it is very easy to build an API around them. Models can be easily retrieved in either XML or JSON format, and it is easy to restrict which data is returned by explicitly defining public data versus private data.

By using some simple URL rewrite rules you can create URLs that are easy to remember, which represent an API endpoint, such as:

```
/users/show.json?username=someuser
```

This API endpoint will return information about the specified user in JSON format. It's also possible to return data in XML format by changing the extension on the URL to xml. Our Android application will communicate primarily through this API.

5.2 Mobile Implementation

The mobile portion of the social network was developed as an Android client. This was done using the Eclipse plugin for the Android SDK. This allows the developer to choose a target API level as well as access emulators for various devices released for that API level. The plugin also allows the developer to use an Android device attached to the computer and instantly load a current version of the project for debugging directly on the device.

5.2.1 Android Versions

When working with Android, the developer is required to choose a target API level. When this project began, the highest API level available was 9 which corresponds with Android version 2.3, also known as Gingerbread. Table 5 shows the distribution of Android versions currently in use as of April 1st, 2011. From table 5 it can be seen that two new API levels were released since the beginning of this project and Android version 2.2 is now the most widely distributed version.

Platform	API Level	Distribution
Android 1.5	3	2.7%
Android 1.6	4	3.5%
Android 2.1	7	27.2%
Android 2.2	8	63.9%
Android 2.3	9	0.8%
Android 2.3.3	10	1.7%
Android 3.0	11	0.2%

Table 5: Android API Distribution (April 1st, 2011)

For the Android application we chose to use API level 7, supporting Android versions 2.1 and up. Since Android versions are backwards compatible, version 2.1 supports roughly 94% of the active Android population. At the beginning of this project this was closer to 75%, which implies that Android versions below 2.1 are quickly becoming obsolete.

5.2.2 Android Development

Android development is java-based with access to API's that support all of the functions of modern smartphones including functions that access GPS location information, accelerometers, direct access to the phone's broadband connection, and much more. Android applications themselves must specify which functions of the phone they will access and users are alerted to these functions before an application is installed. If an application attempts to access a function that it previously hasn't indicated intent to use, the application simply stops working and closes. The Campustream application uses the "Internet" permission as its only permission meaning users are notified that the application can access the web through either Wi-Fi or through their mobile provider's network.

The layout of Android applications is written in XML and each layout belongs to an activity within the application. An activity is essentially a screen of the application that is opened. For example, the login page of Campustream is one activity and when a user presses login, it calls another activity to open the stream page of the network. This makes allowing the user to return to previously viewed pages quite easy, because pressing the back button on an Android phone simply closes the activity on the top of the stack, essentially opening the previous page the user was on.

Activities must be pre-defined similar to permissions. This is done in an XML file called the AndroidManifest. This contains the activities and permissions the application uses. When activities are defined they can be given additional characteristics that allow the developer to define what orientations the activity can use, what status bars or title bars to display along with the activity (i.e. A bar with the title of the application), as well as what type of activity it actually is. The AndroidManifest file also contains the current version of the application, as well as its target API level so that when the application is uploaded to the Android Market the Market knows which devices are capable of using the application.

As previously stated the development itself takes place in Java, which calls upon XML files at the creation of each activity. Each class file in the Eclipse project represents either an activity or a set of APIs or objects that a number of activities call. The majority of the activities in the application are listViews. These are views that allow content to be added or removed in a list-based form. Android automatically creates scrollability options for listViews so when new content is added; the application adjusts itself to support it. This works well with a social network because users post their own content and each item has identical characteristics that need to be displayed.

5.2.3 Accessing Web APIs

All the content that the Android application pulled in was accessed through a set of APIs from the web, listed in Appendix C. These APIs return information in either Json or XML format and then the Android client uses the standard Json or XML parsers to retrieve the information from the returned string.

One challenge our project had to overcome was the loading of icons and images to be used in the listViews through the application. The data displayed is recovered on a per-item basis. This means that each item in the list is its own object with no link to other objects, so the application does not inherently know which listView items are similar or the same as other items. This presented a challenge in that the application would download a user's profile image a significant number of times if they were an active poster, when really it would only need to download the image once.

This was resolved by storing each image downloaded from any section of the application in a universal image cache. This way a user's icon could be downloaded on the Stream and the application wouldn't need to download the icon again if that user posted in the Collaboration

section or any other section. This greatly improved performance but still had issues as long lists would all be downloaded at once.

This was resolved by using "Lazy Loading". This is when only the images that are actually being viewed by the user are actually downloaded. As the user scrolls through the list, the images are downloaded on the fly, as they are needed. This keeps the maximum number of images being downloaded at any given time to around four or five and greatly reduces the strain put on the mobile network connection. To further relieve this strain, images are cached in between uses of the application, meaning that images that were downloaded on the last use will not need to be downloaded again. All of these techniques combined made the user's experience much smoother than before they were implemented.

5.2.4 Android Market

The application is exported from Eclipse as an APK file that can then be uploaded to the Android market. This process is done by using a keystore given to each registered Android developer, which requires a password to access. This is done to prevent unwanted copying of the application onto unauthorized devices. After the application is exported the developer uses the Android Market Publishing Console to publish the application. This requires uploading the APK, assigning a name, description, content rating, and category. Campustream can be found under the "Social" category of the android market and is rated appropriate for all ages. The developer is also allowed to upload various marketing icons that are used in different portions of the Android Market application as well as up to eight screenshots highlighting the application. After the application is submitted it appears on the market almost instantly for all supported devices. This is very different from the Apple AppStore and the BlackBerry App World where a long, multi-week approval process is required. Screenshots of the submission process for Android applications can be found in Appendix D.

7 Results

The original plan was to release Campustream to the entire undergraduate WPI community after some pilot testing with friends. However, due to complications with emailing the entire undergraduate list, Campustream was announced to students in three batches. The first mass-email was sent to several dormitories and fraternities, as well as the Frisbee team. This reached roughly 500 students and occurred on March 21st, 2011. The second mass-email was sent to the CS and IMGD undergraduate majors at WPI. This added roughly 500 additional students to the reach of the project and occurred on March 27th, 2011. Finally the entire undergraduate student list was emailed, reaching 3000-4000 students, occurring on April 7th, 2011.

As a further effort to drive traffic onto Campustream, we announced a gift-card giveaway on April 14th, 2011. This was done by sending an email to everyone registered for Campustream and letting them know that every time they posted they would be entered to win a 20$ Amazon gift card. This content ran until April 19th, 2011, when a winner was finally chosen. Project presentation day occurred on April 21st, 2011, which could have also impacted our results as user interest may have risen.

As of April 21st, 2011, Campustream has reached 209 unique user accounts all linked to WPI email addresses. The number of unique users per day is averaging in the low double-digits and there have been over 1000 visits to Campustream.com itself. The following graphs show more specific data collected by Google Analytics, Android Market Analytics, or SQL queries done on the Campustream database.

7.1 Pre-Use Survey

To get a good sense of user behaviors and attitudes towards social events and social networking, we surveyed registered users as part of the registration process. We believed that gathering this data beforehand, would give us a good baseline to judge the impact of our application. Our survey asked users several questions about their event attendance, the ease of finding out information about information and events related to campus, and the effectiveness of MyWPI. Over 120 people answered the survey and the results are as follows.

7.1.1 Social Event Attendance

Users were asked, "How many of each of the following types of social events do you attend each month of average?" As figure 14 shows, the most commonly attended events were "Parties or gatherings" followed by "Other events". Unfortunately we did not ask the user to define what the "Other" actually was. From this, we concluded that events hosted by WPI are in the minority and that users attend more events simply hosted off campus or spending time with their friends.

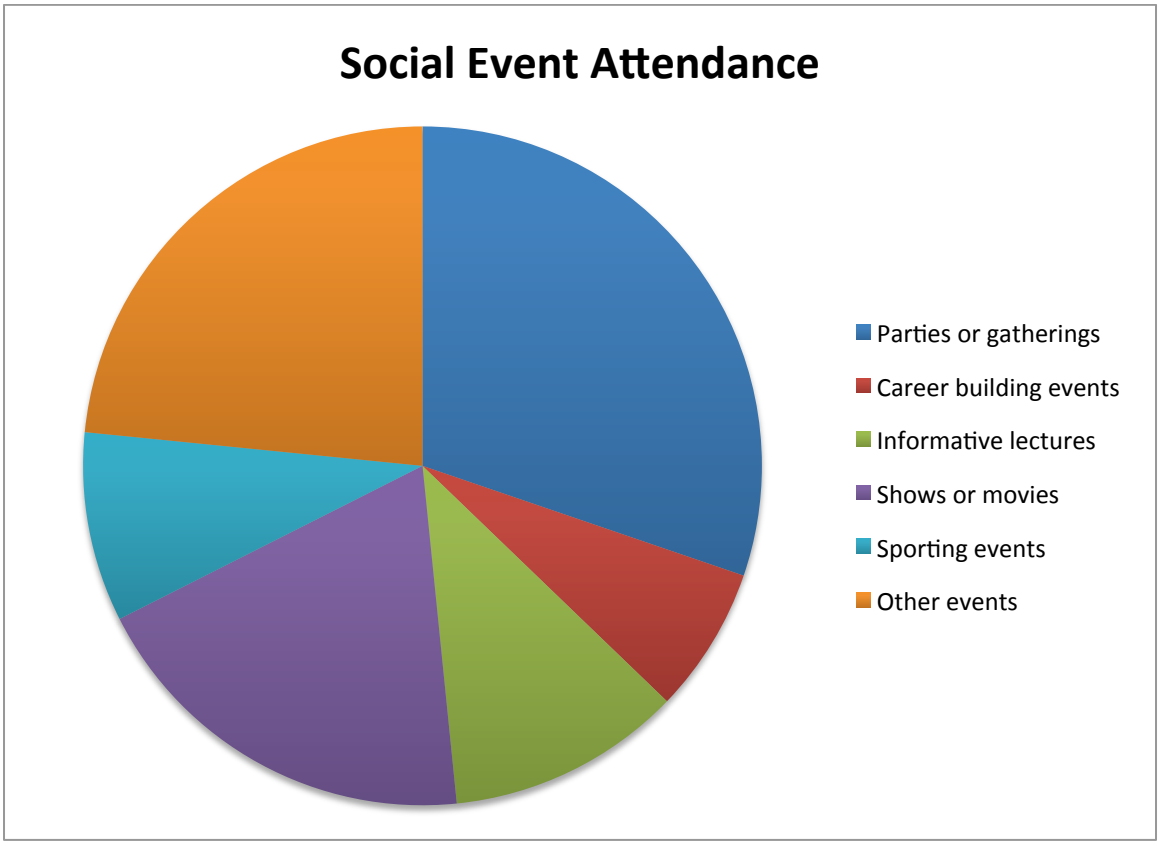


Figure 14: Social Event Attendance

7.1.2 Ease of Finding Information about Social Events

Users were asked, "On a scale of 1 to 10, how easy is it to find answers to questions you have related to social events or activities". Most users found it relatively easy to find out information about social events. However there was a significant portion of users that were neutral about the topic, showing that there is a portion of students that would gladly accept an easier approach.

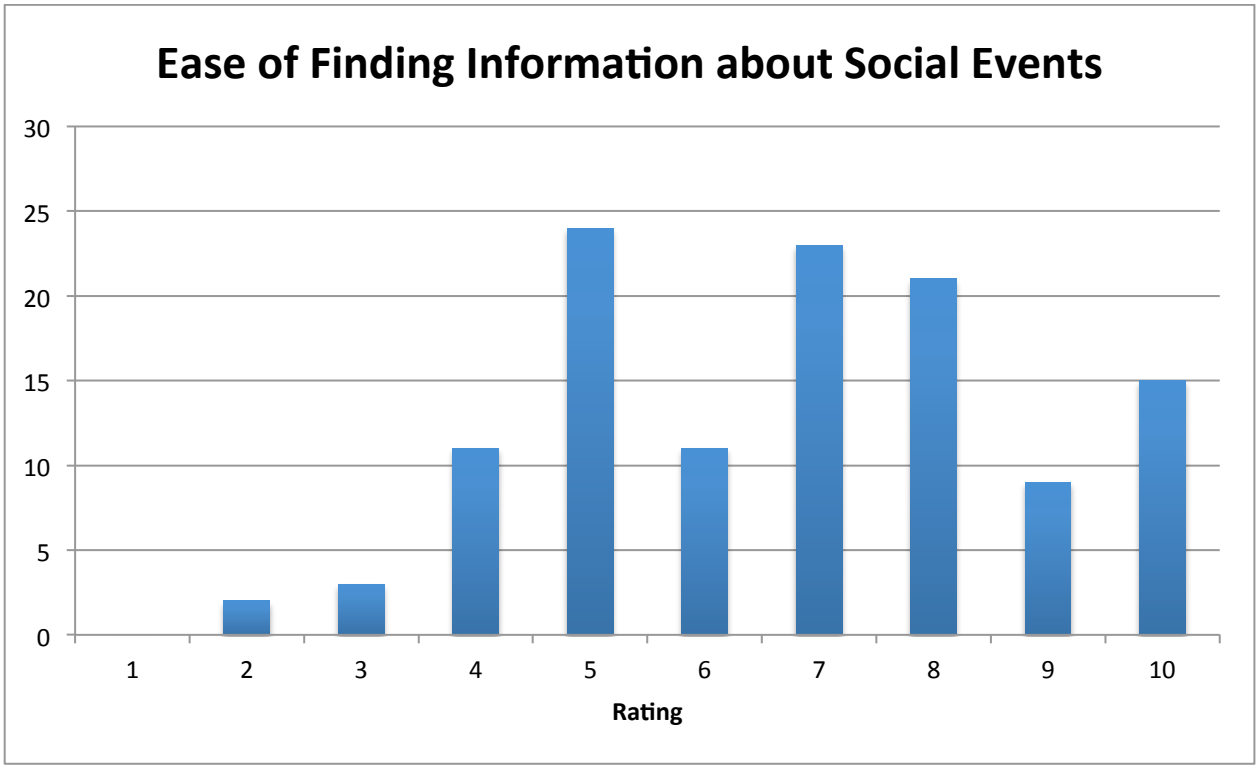


Figure 15: Ease of Finding Information about Social Events

7.1.3 Ease of Finding Class Due Dates

Users were asked, "On a scale of 1 to 10, how easy is it to find answers to questions you have related to class due dates or information". The large majority of users found it very easy to find information related to classes, including due dates. This shows that most users aren't hassled by finding this information. Although, a small portion of users seem to be having a difficult time finding class related information, showing that users certainly wouldn't reject a new method of doing so.

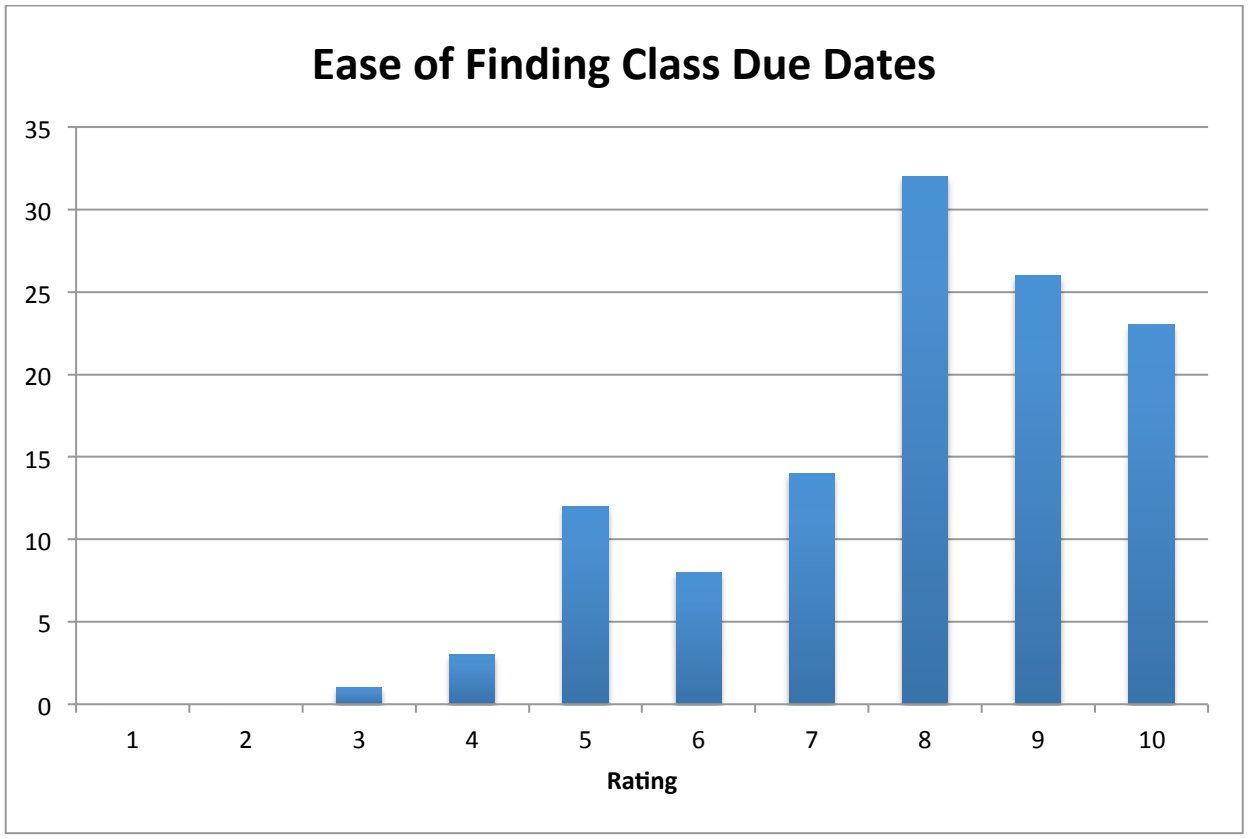


Figure 16: Ease of Finding Class Due Dates

7.1.4 Ease of Finding Information about School

Users were asked, "On a scale of 1 to 10, how easy is it to find answers to questions you have related to miscellaneous school-related questions". The answers for this question widely varied, with the majority of users being relatively neutral about the answer. This means users are having difficulty finding information related to WPI that isn't available through their classes.

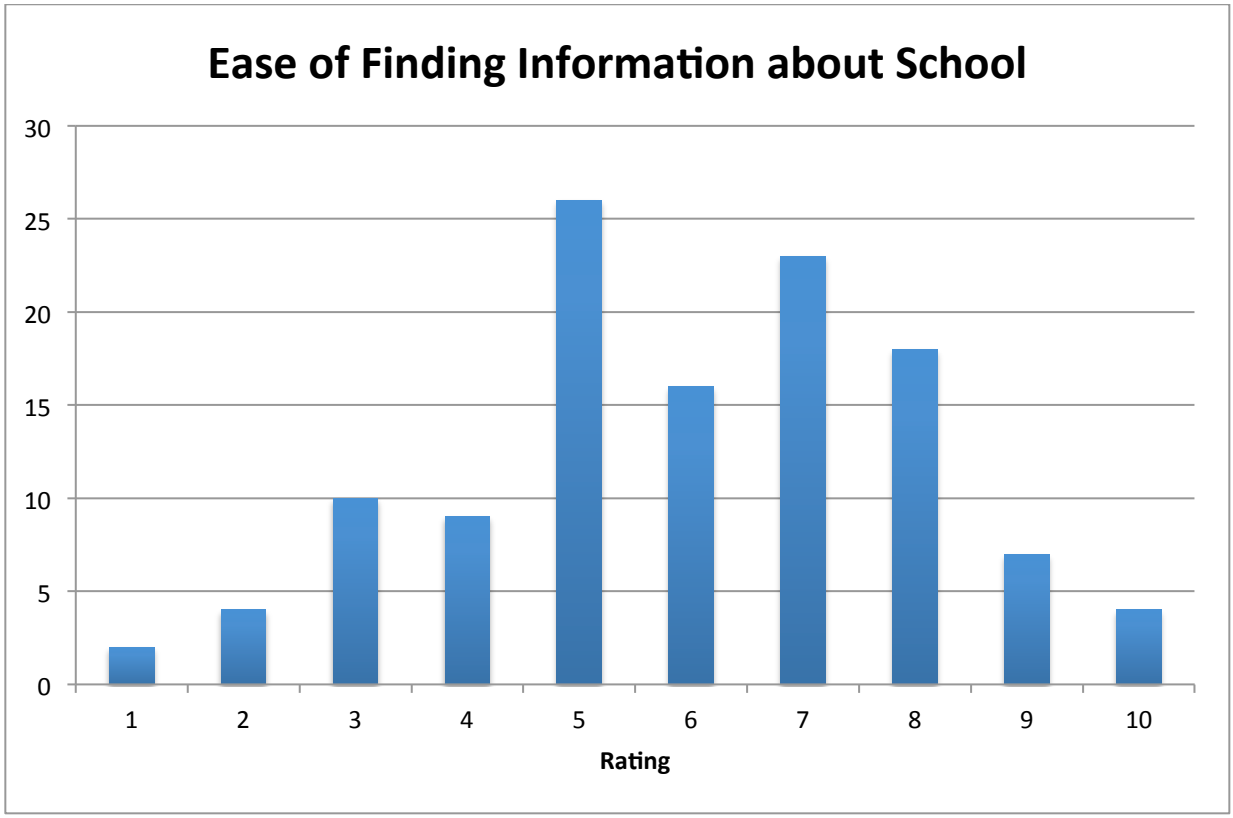


Figure 17: Ease of Finding Information about School

7.1.5 Ease of Finding Information about Security Protocols

Users were asked, "On a scale of 1 to 10, how easy is it to find answers to questions you have related to safety information or security protocols". The results to this question are worrying in that there is a significant portion of users that have a very hard time finding information about security on campus. Security is something that WPI takes very seriously and the results of this question show that WPI authorities should consider additional security measures or alternate ways to disseminate security-related information.

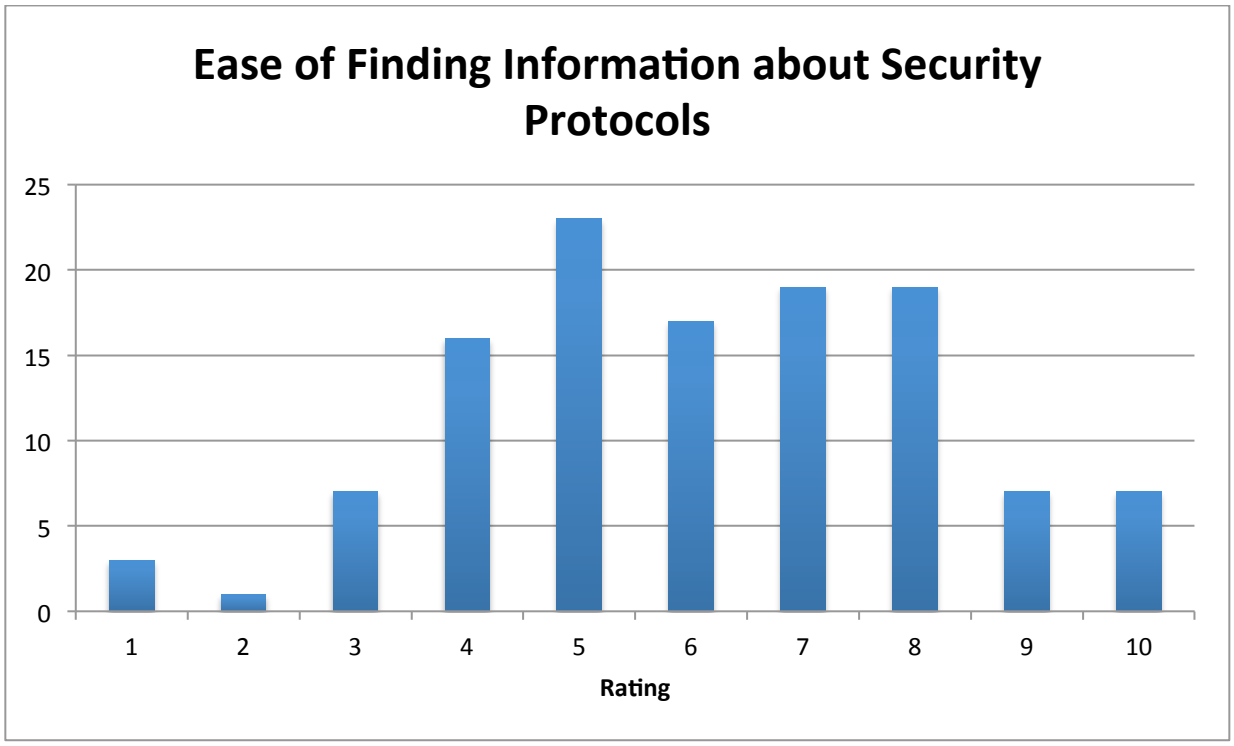


Figure 18: Ease of Finding Information about Security Protocols

7.1.6 How Often is MyWPI Used when Available

Users were asked, "How often do you use the discussion boards on MyWPI when they are available for class?" The majority of users either never used the discussion boards or rarely used them. This mimics the results of our pre-survey where users said they found MyWPI to be less than useful and that an alternative, more open message board would be something they would like to see.

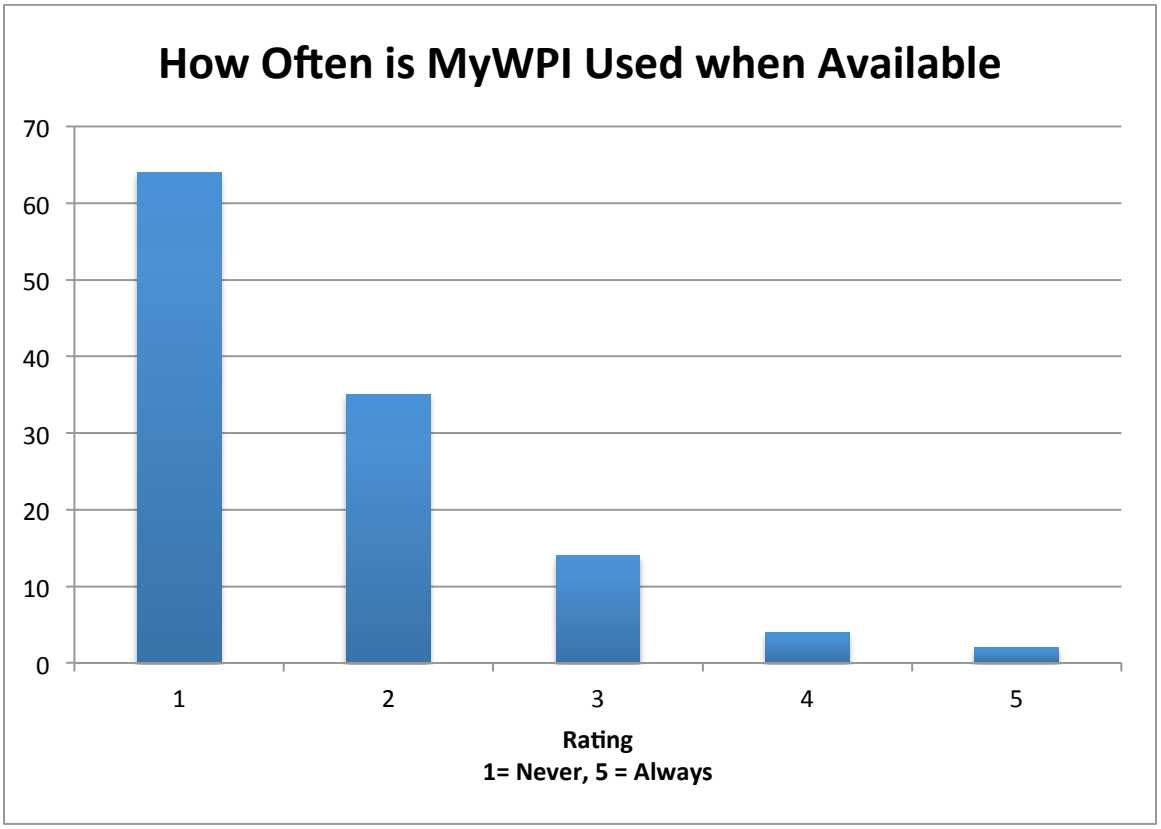


Figure 19: How Often is MyWPI Used when Available

7.1.7 Effectiveness of MyWPI Discussion Boards

Users were asked, "How effective do you feel the MyWPI discussion boards are in answering questions you have about a class?" The results for this question are highly correlated with the previous question about how often users actually use the MyWPI message boards. It makes sense that because users rarely find the message boards effective that they would be dissuaded from using them at all.

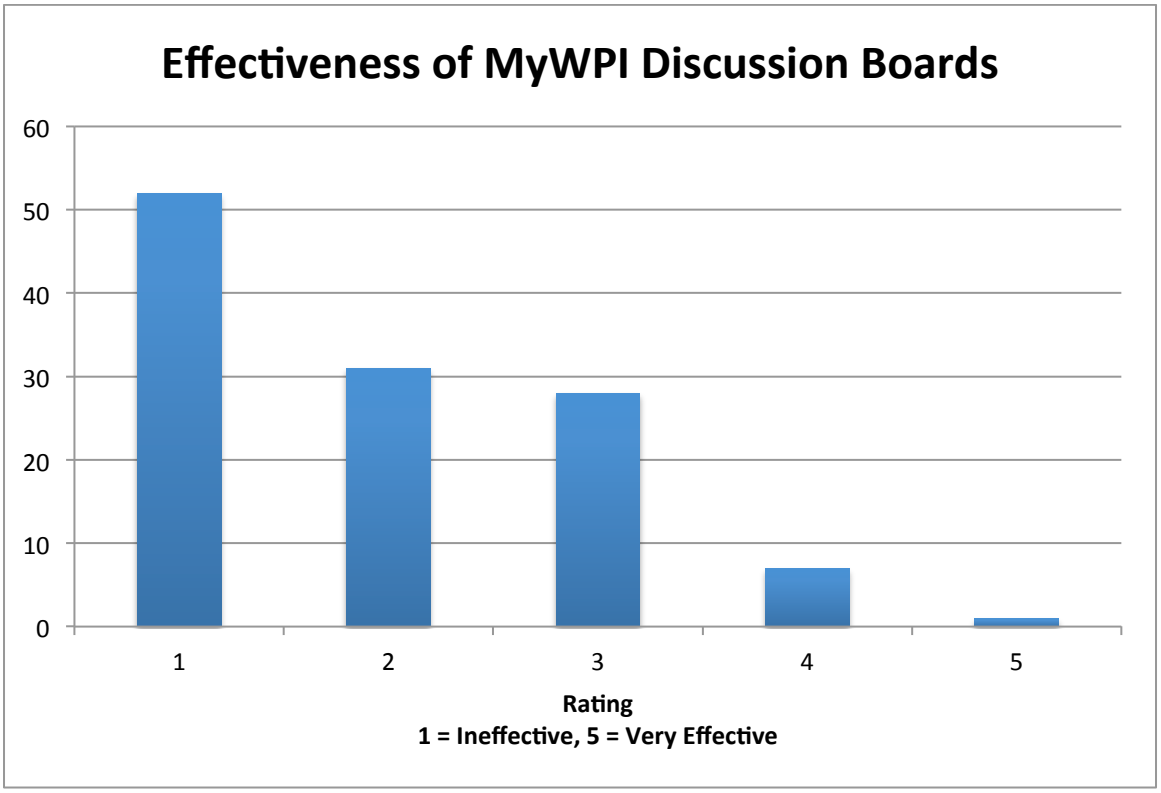


Figure 20: Effectiveness of MyWPI Discussion Boards

7.1.8 Witnessed Suspicious Activity on Campus

Users were asked "Have you ever seen any suspicious activity on campus and either didn't know where to report it or found it difficult to do so?" The results of this question are also worrying as roughly 20% of respondents answered, "Yes". This coupled with the previous security related questions shows that security protocols need to be made clearer or a new method of submitting suspicious activity needs to be developed.

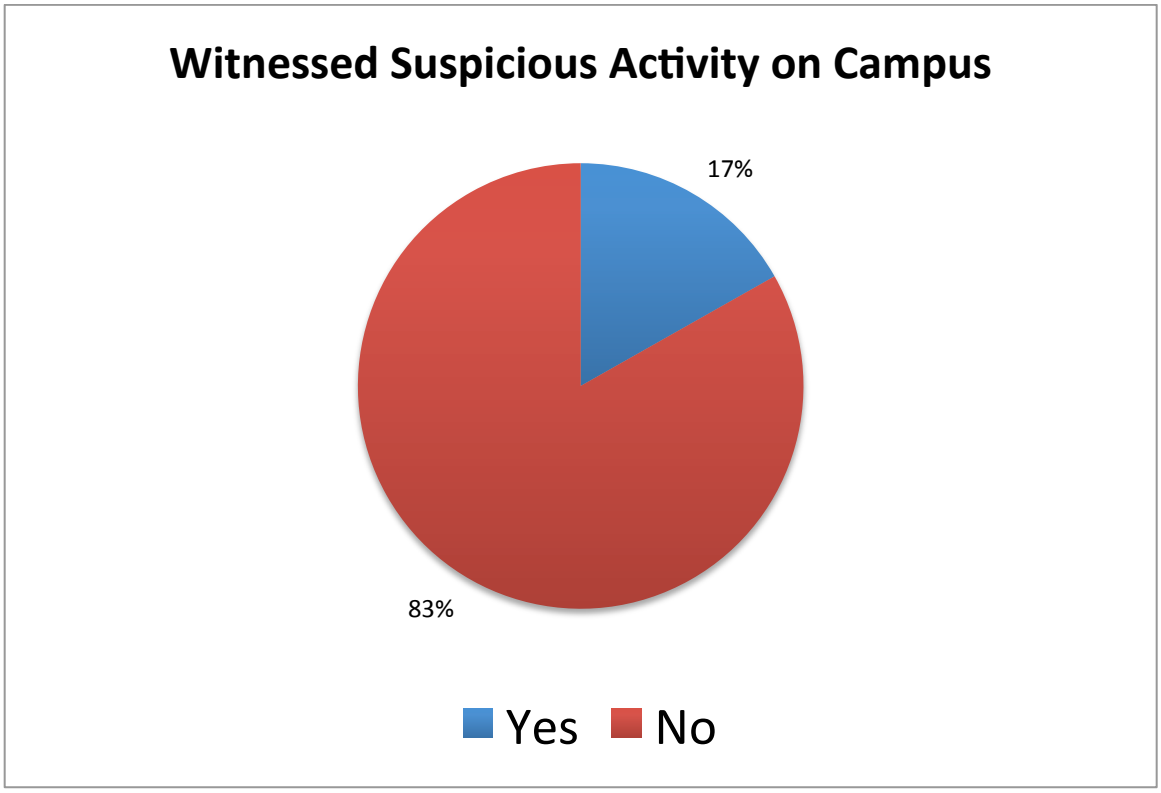


Figure 21: Witnessed Suspicious Activity on Campus

7.1.9 Want to Know More Information about Events

Users were asked "Have you ever wanted more information about an event on campus and didn't know how or where to find more information?" The majority of users answered "Yes" to this question, showing that students would like to see a central place or some better way to find more information about events on campus.

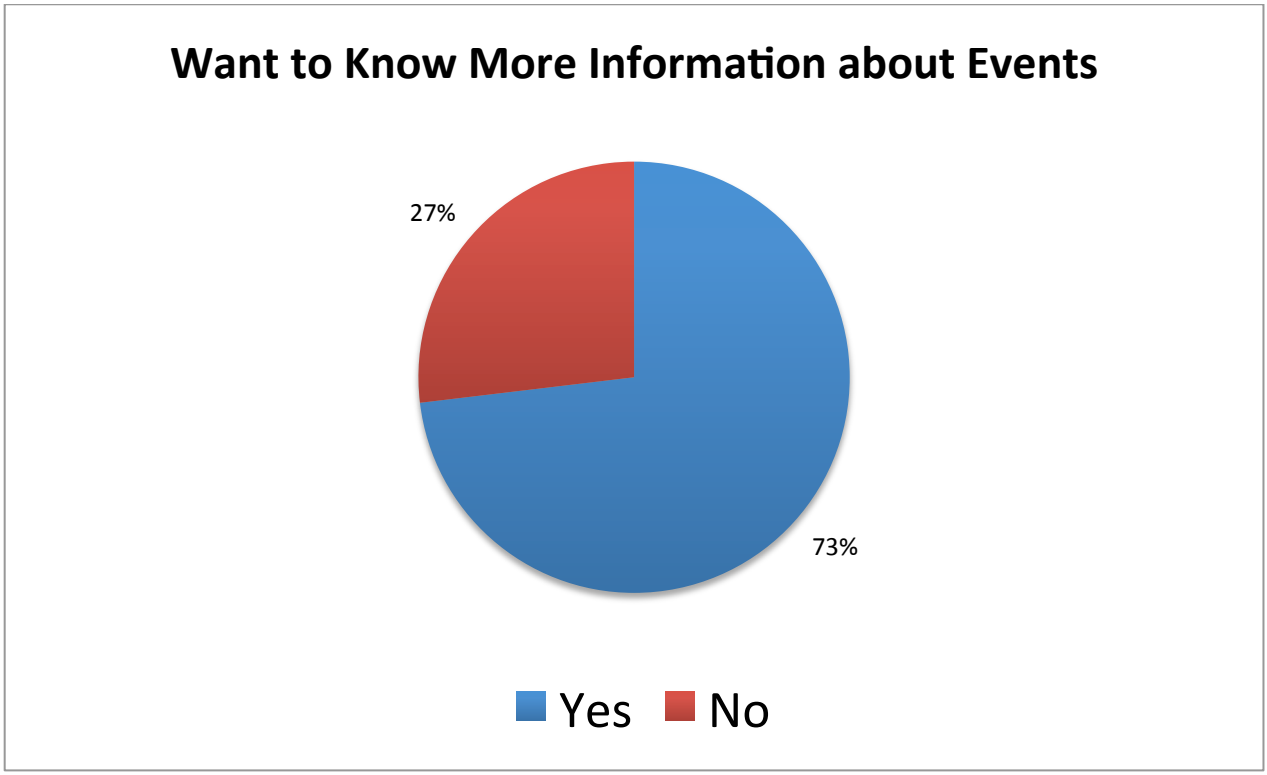


Figure 22: Want to Know More Information about Events

7.2 Unique Visitors VS New Accounts

This graph shows the number of unique visitors to the website each day as well as the number of new accounts activated each day. Three distinct spikes are visible in both statistics, which correspond to the dates on which the three mass-emails were sent out. The first two mass-emails were to roughly 500 students each while the final mass-email reached over 3000 students. The graph below shows the resulting data from March 14th, 2011 to April 23rd, 2011.

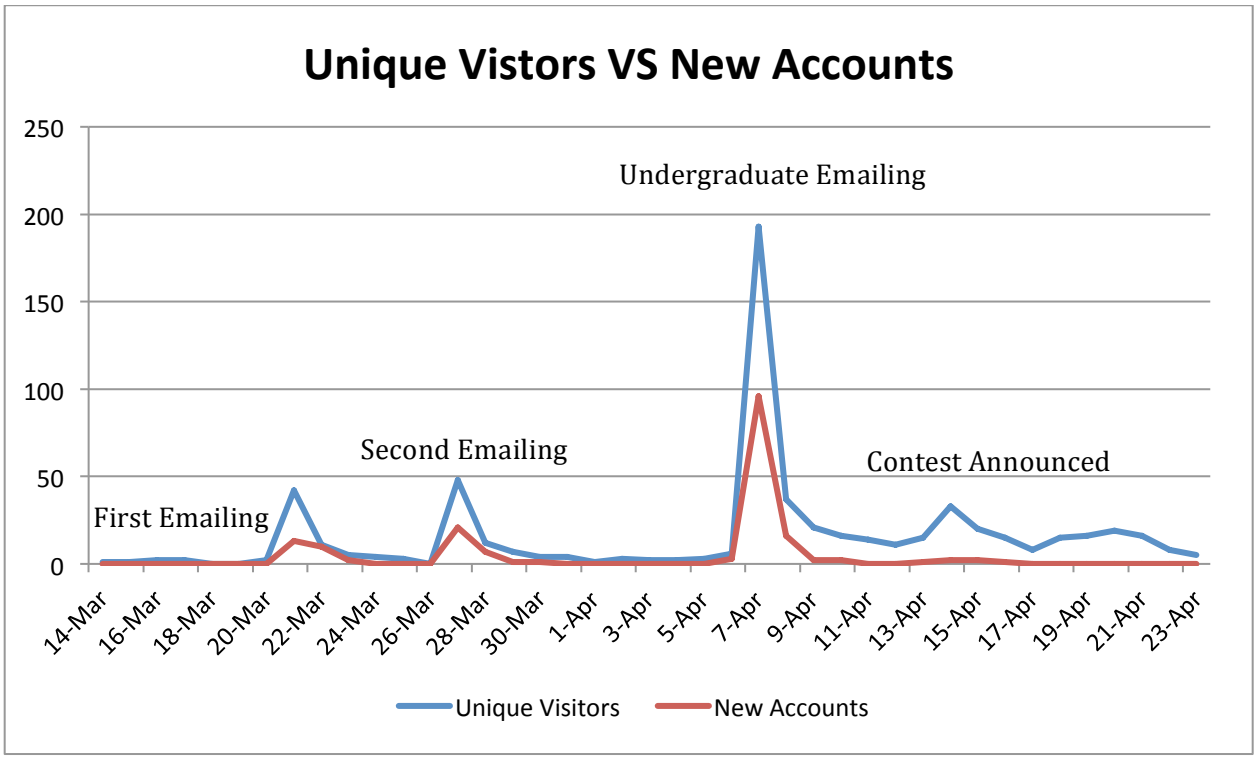


Figure 23: Unique Visitors VS New Accounts

7.3 Visits per User

This graph shows the number of times each unique user visited the website. This shows both unregistered and registered users combined. As you can see the majority of users, (over 34%), only visited the website a single time. However, a significant portion of users, over 38%, visited the site more than nine times. The graph below shows the resulting data from March 14th, 2011 to April 23rd, 2011.

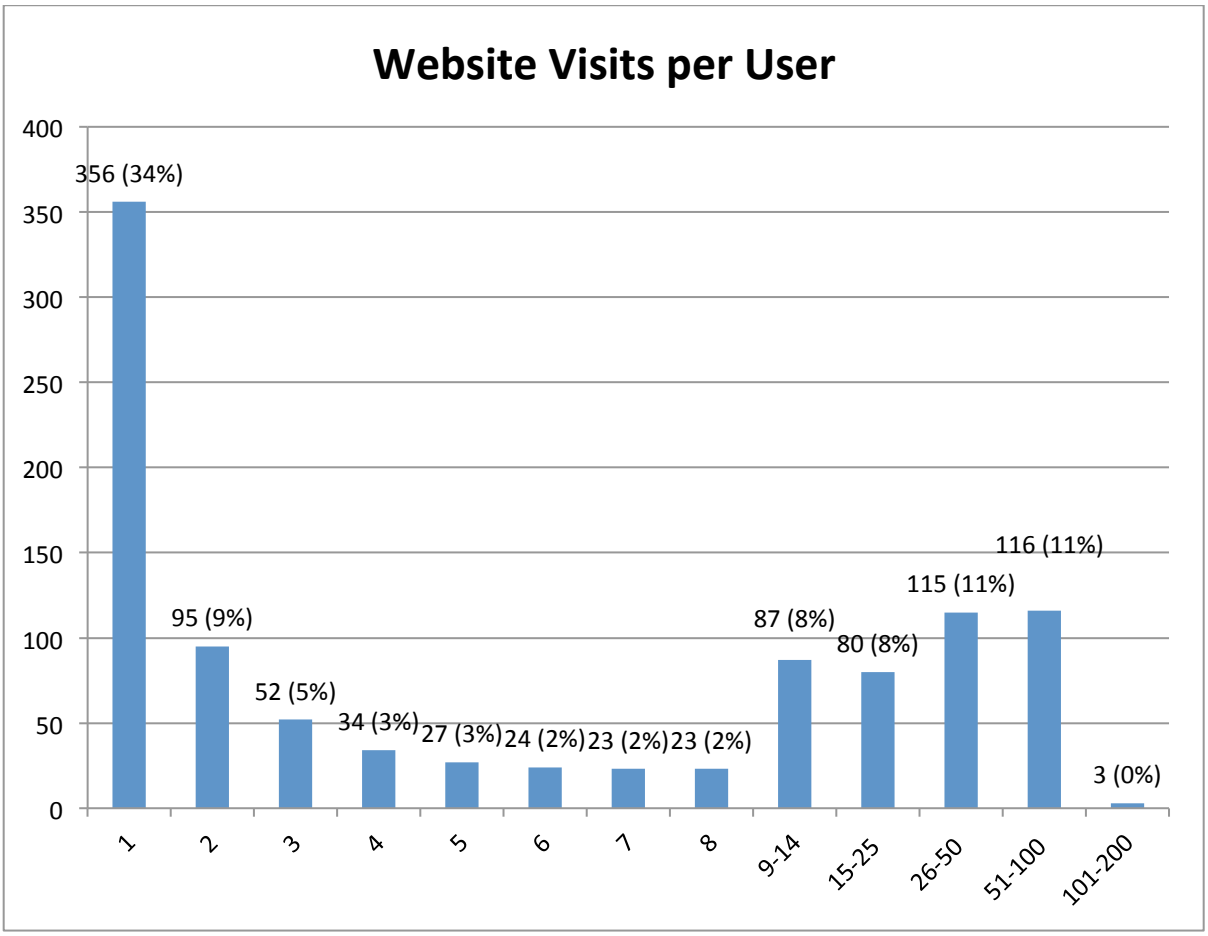


Figure 24: Website Visits per User

7.4 Time between Visits

This graph shows how long a user waited in between visits to the website. As you can see a significant portion of the visits to the website were actually first time visits. The majority of return visits also occurred the same day as the user's initial visit. Only 98 users returned to the website after not visiting for more than a day, this represents about 10% of users in total. The graph below shows the resulting data from March 14th, 2011 to April 23rd, 2011.

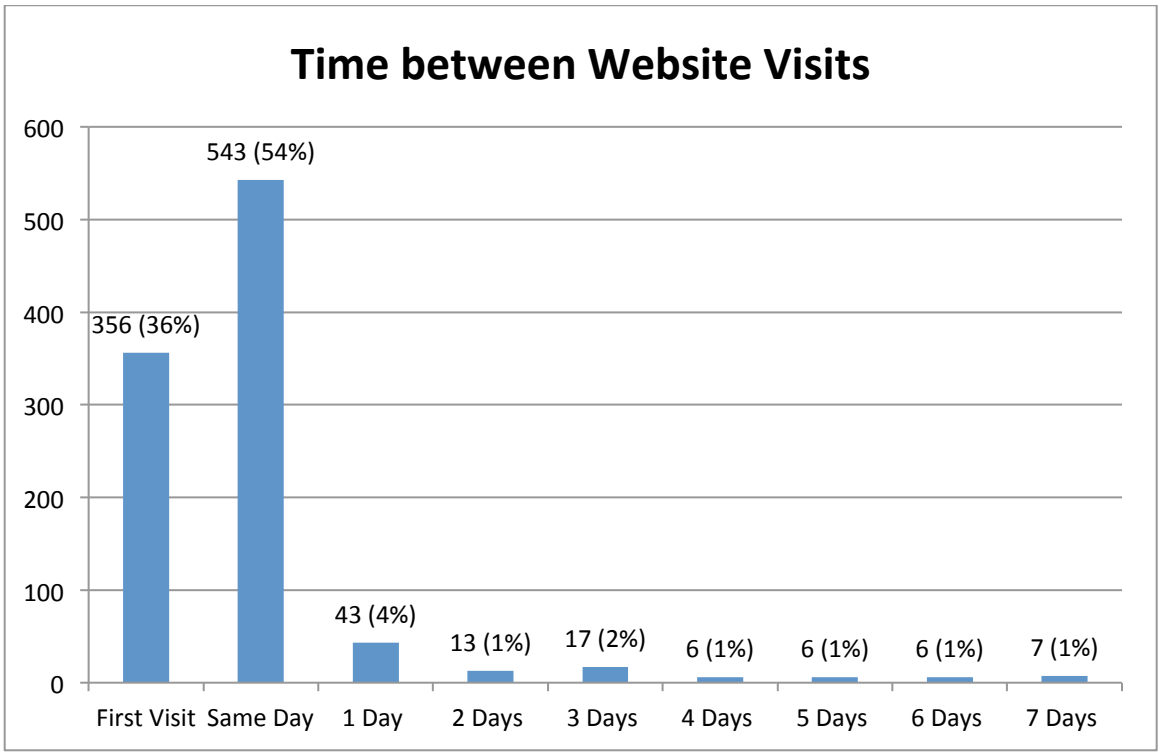


Figure 25: Time between Website Visits

7.5 Active Android Application Installations

This graph represents the number of active application installations over time. This graph shares the same three distinct spikes present in several other graphs corresponding to dates on which the mass-emails were sent out. While there are 209 total Campustream accounts, only 26 users have the dedicated application currently installed on their Android phone. The application itself has over 120 downloads, however this shows either that Android users that didn't understand what the application was and simply downloaded it by mistake or there were users who uninstalled the application after use. The number of active installations peaked at 31 shortly after the entire undergraduate list was emailed and has since dropped off slightly. The graph below shows the resulting data from March 14th, 2011 to April 23rd, 2011.

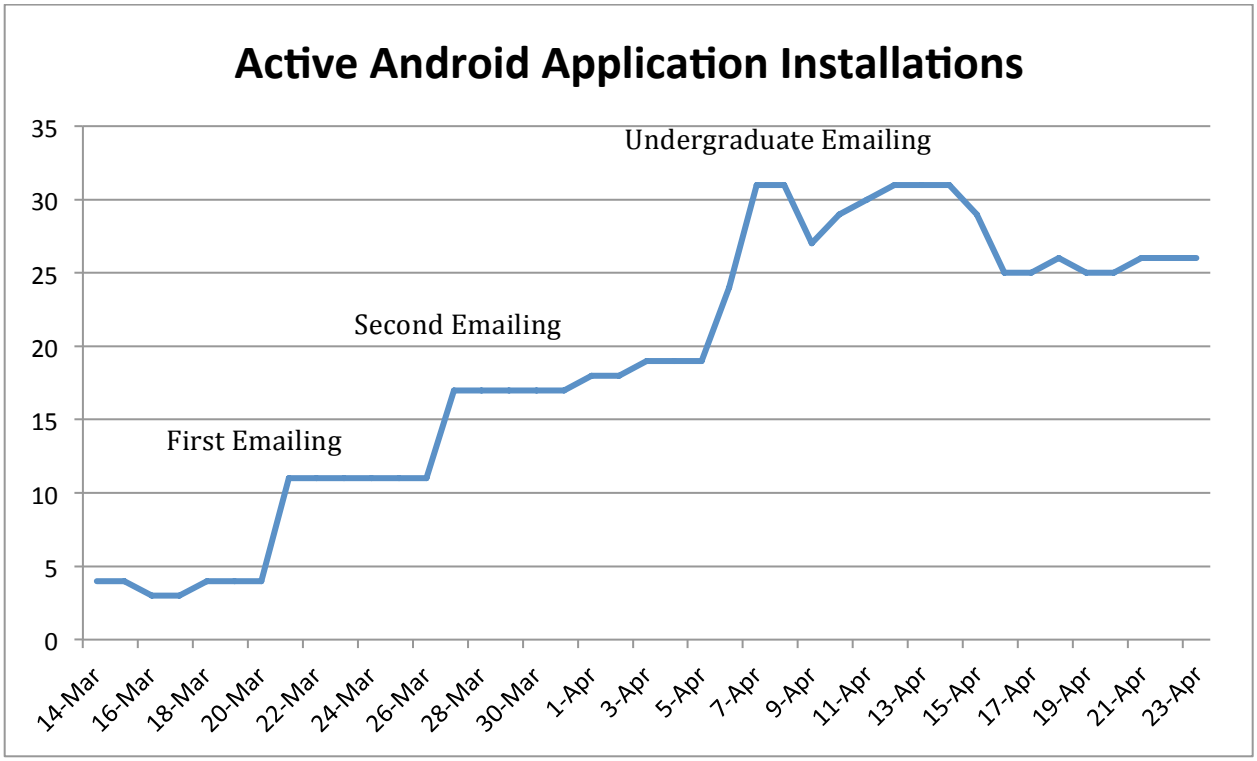


Figure 26: Active Android Application Installations

7.6 Application Installs by Android OS Version

The Android application only supports Android versions 2.1 and higher. Android 2.1 and higher currently represents over 90% of Android phones and this number is increasing every day. As you can see the large majority of active installations are from Android 2.2, which makes sense from a more technical school. This information was retrieved from Google's Android market statistics provided to developers. Google provides a breakdown of downloads by device type as well as Android version to allow developers to see which devices or versions should be supported. The graph below shows the resulting data from March 14th, 2011 to April 23rd, 2011.

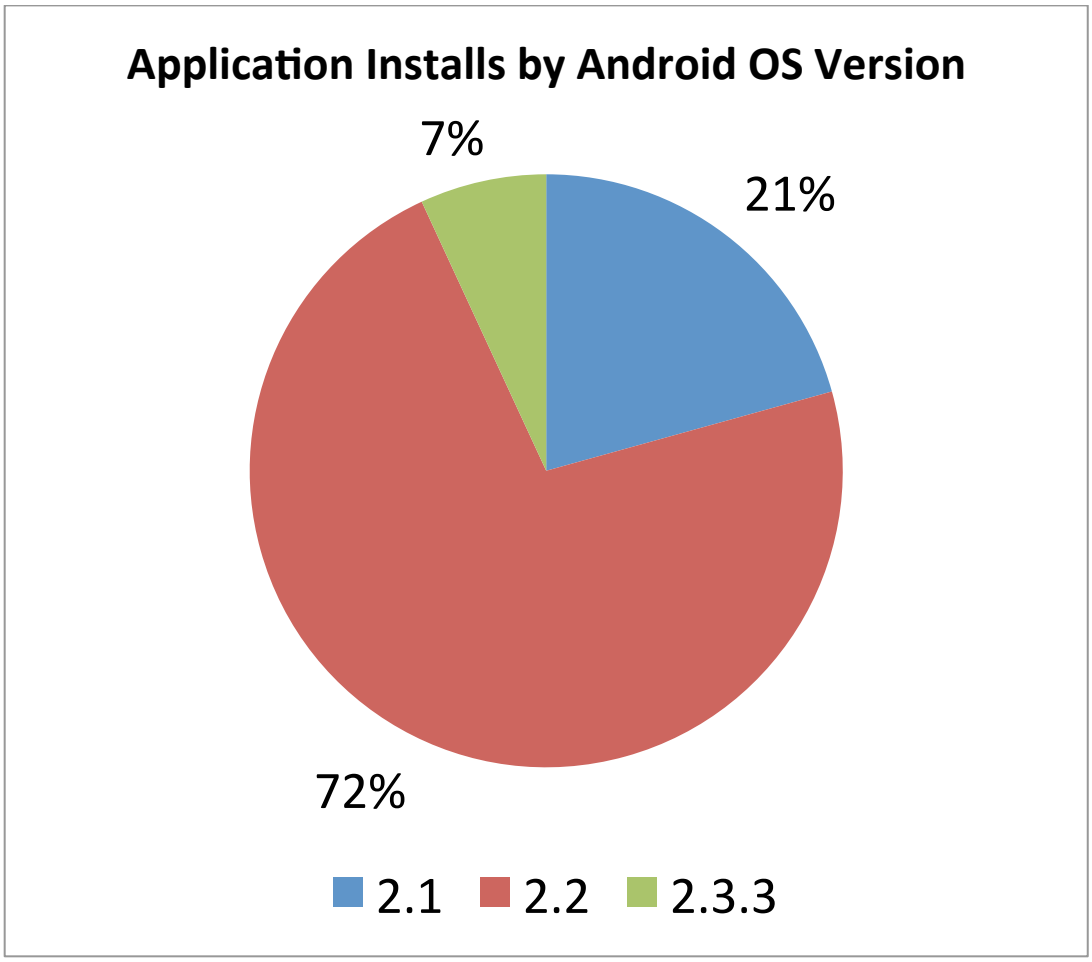


Figure 27: Application Installs by Android OS Version

7.7 Mobile Website Visits by Device

This graph represents the number of mobile visits to the website since launch. Despite having a dedicated Android application, one third of the website's mobile visitors were using an Android phone. This is possibly due to lack of knowledge of the application, or the fact that users could not register from within the application and needed to visit the website in order to create an account before using the application. The remaining two thirds of visits were from iOS devices including the iPhone, iPod, and iPad. There were no mobile visits from other types of devices. The graph below shows the resulting data from March 14th, 2011 to April 23rd, 2011.

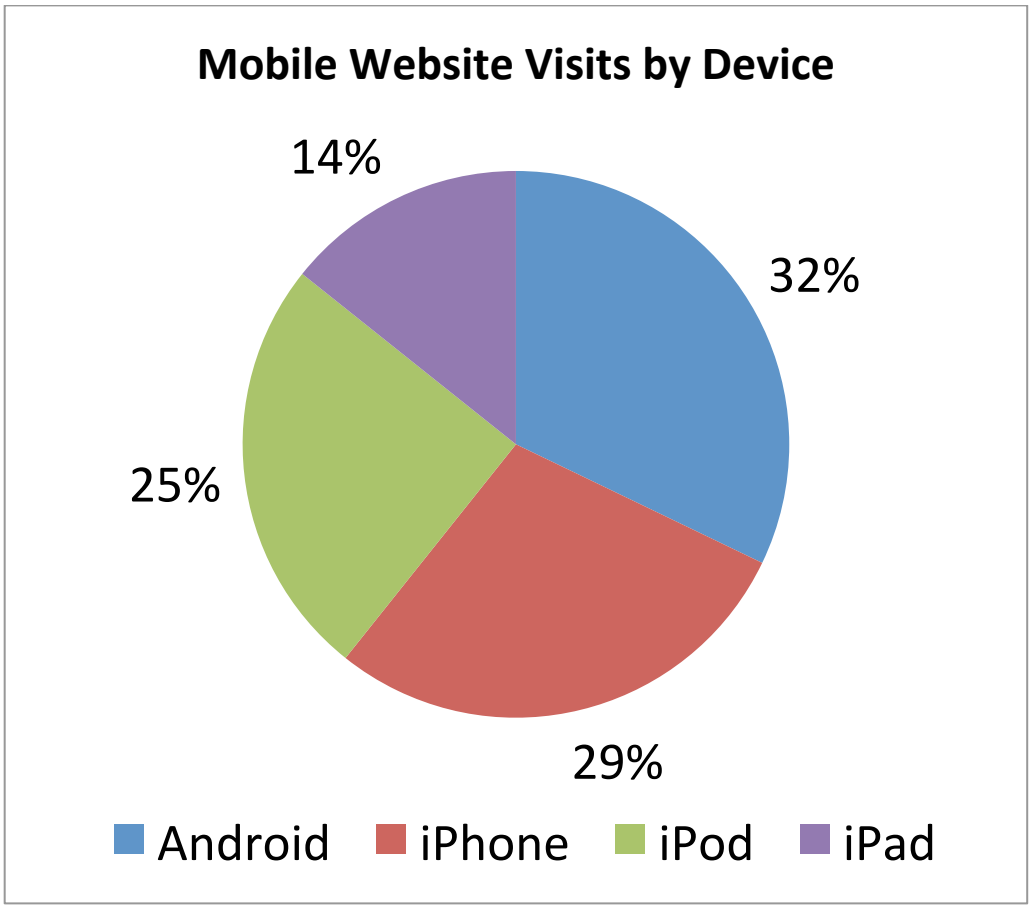


Figure 28: Mobile Website Visits by Device

7.8 Status Updates over Time

This is a timeline view of how many statuses were posted each day on the network. Some dates are omitted because no updates were posted on those days. As you can see, the number of updates very closely correlates to the number of unique visitors and the number of new accounts each day. You can also clearly see the three major spikes from the mass-emails on this graph as well. The graph below shows the resulting data from March 14th, 2011 to April 13th, 2011.

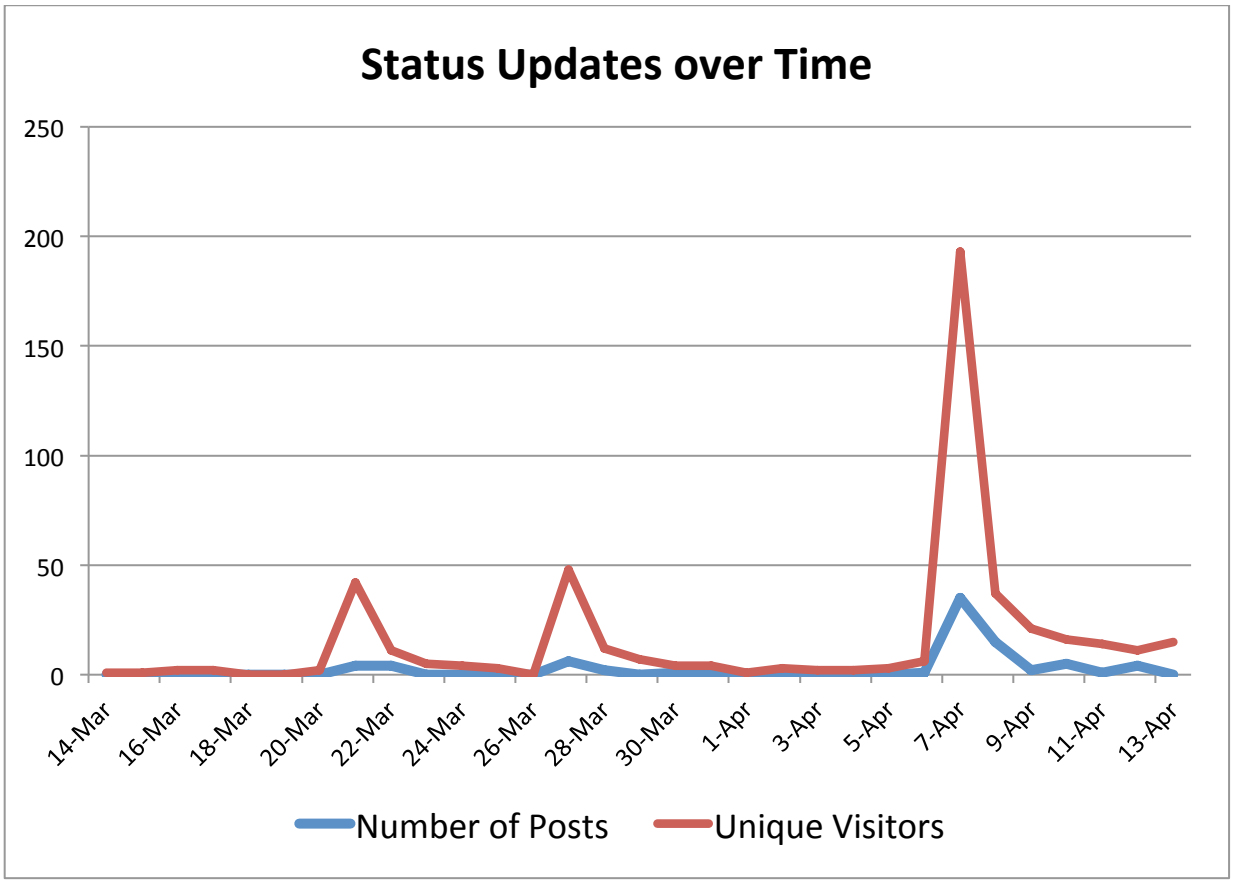


Figure 29: Status Updates over Time

7.9 Time Distribution of Site Postings

When various posting times are grouped by time of day, we can see when students are the most active on average. There were no posts between 8am and 11am, but every other hour had at least one post on the site. Strangely, there was a decent amount of activity late at night, even at 3 – 4am. By far, the busiest period of the day was the afternoon around 2 – 3pm.

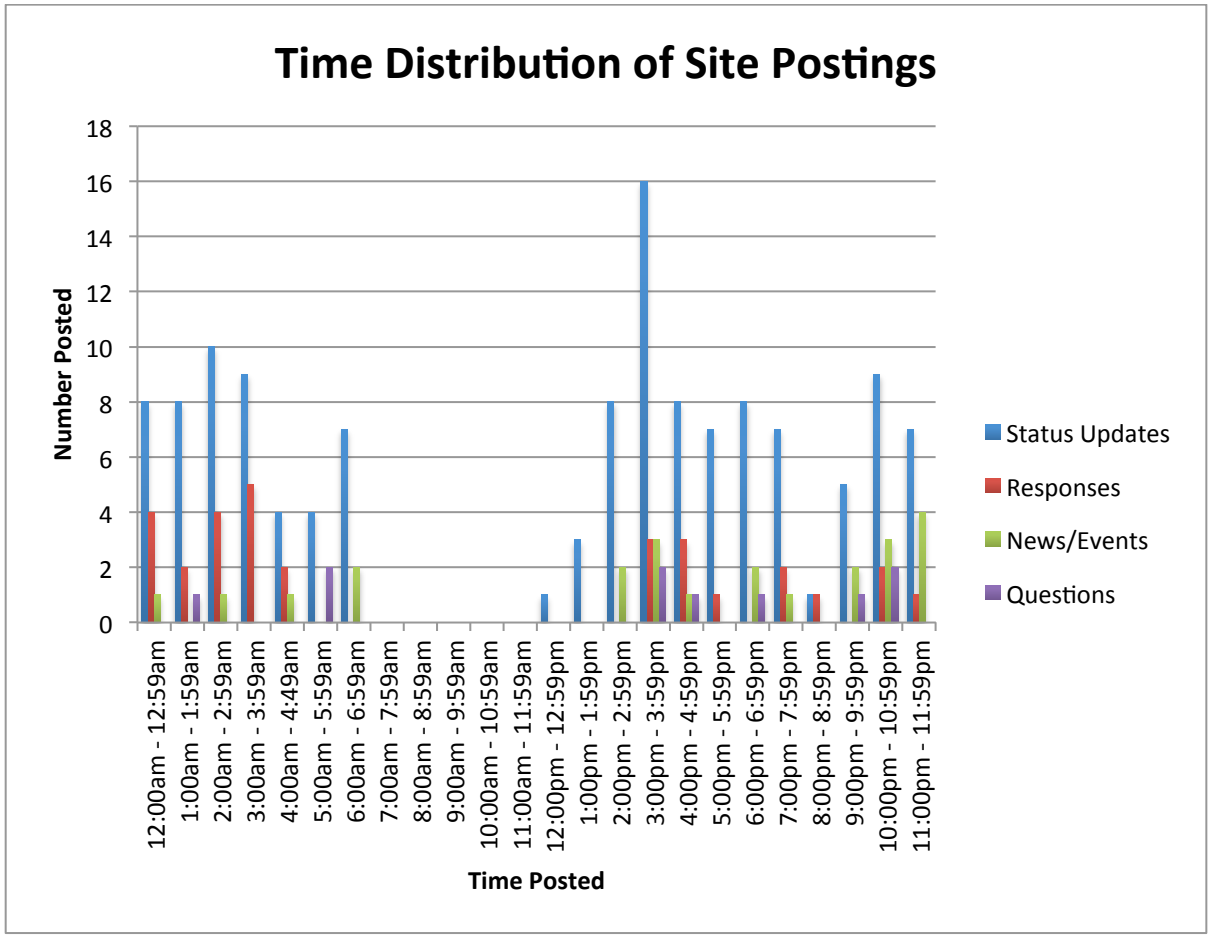


Figure 30: Time Distribution of Site Postings

7.10 Types of Statuses

This graph shows a breakdown of the types of status messages posted to the network. This includes text statuses which are simply status postings on the main stream of the website, question postings which are posted in the collaboration section, responses to question postings, and news or event postings. The majority of messages are simple text postings on the stream of the website, showing that besides the main page, users rarely ventured to other parts of the website. The graph below shows the resulting data from March 14th, 2011 to April 13th, 2011.

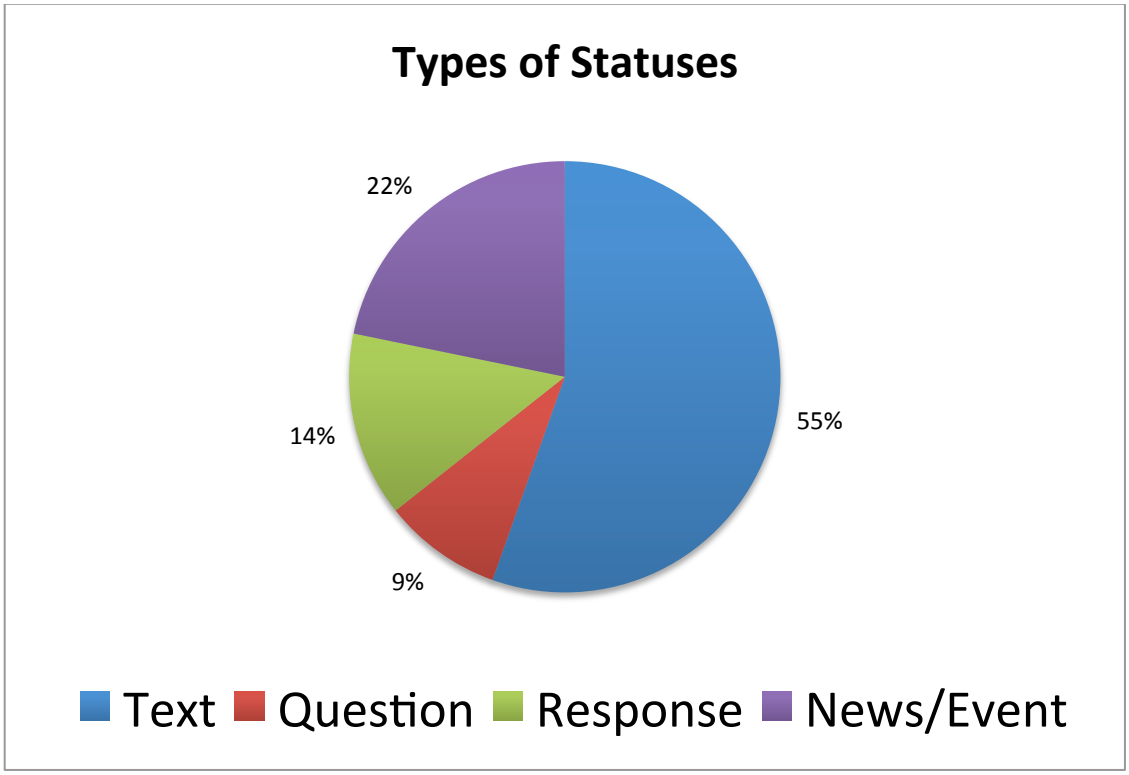


Figure 31: Types of Statuses

7.11 Number of Comments per Status

This graph shows the number of comments per text status. These are the status updates posted to the stream portion of the website. Figure 32 shows that the majority of statuses received two or fewer comments, while only a few status updates received a higher number of comments. The graph below shows the resulting data from March 14th, 2011 to April 13th, 2011.

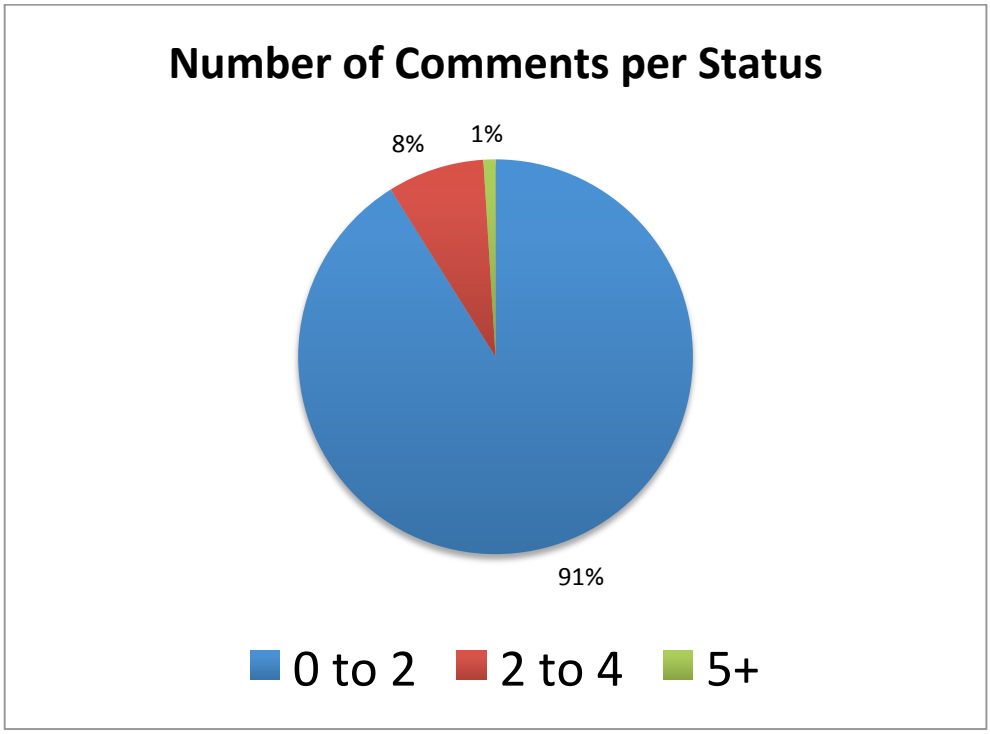


Figure 32: Number of Comments per Status

7.12 Average Engagement

The average engagement is calculated by finding the ratio of follow-ups to content postings. For example, for each type of message, the percentage of engagement is calculated by dividing the total number of responses by the total number of messages of that type. This allows us to see how active each section of the site is after normalization. The collaboration section of the website where students are able to ask questions about events or academics received the largest proportion of engagement. In fact, on average each question received three or more responses. The graph below shows the resulting data from March 14th, 2011 to April 13th, 2011.

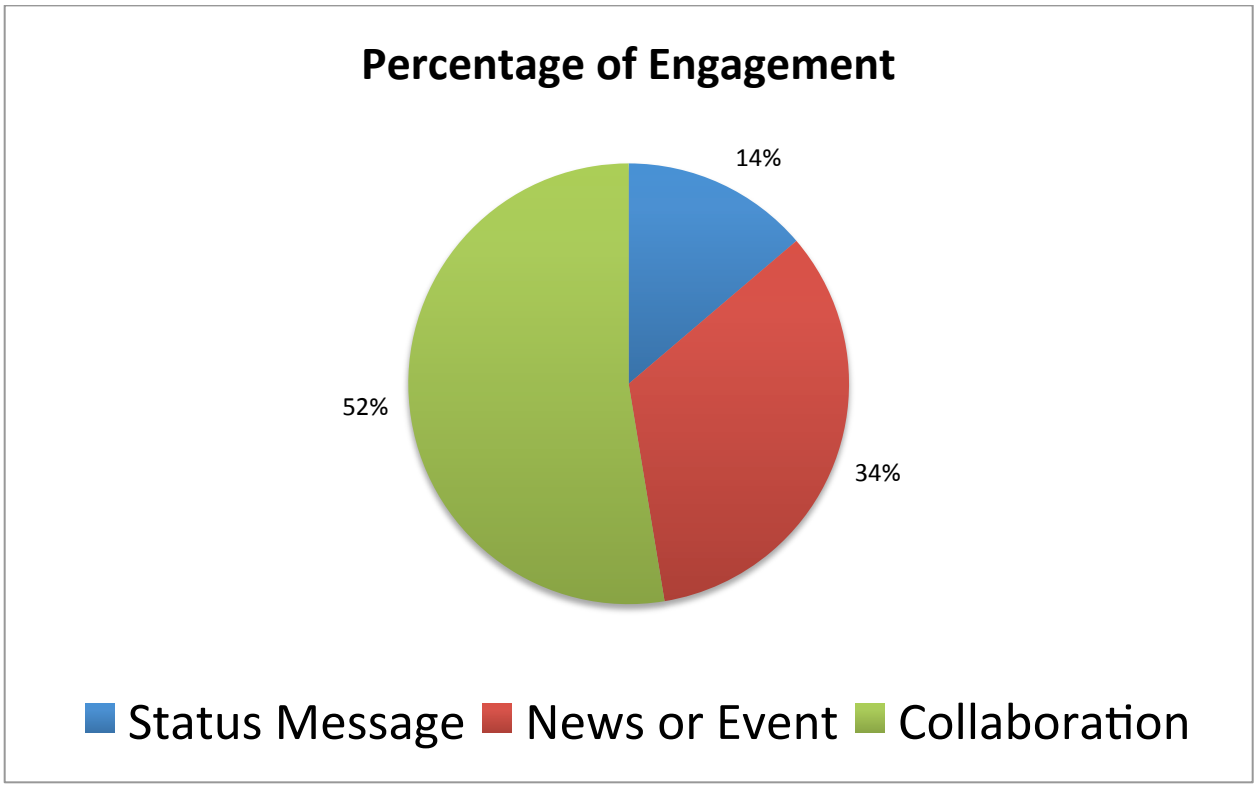


Figure 33: Percentage of Engagement

7.13 Posts VS Followers

This is a scatter plot that relates the number of followers of a user to the number of status updates that user has. This is useful because it shows whether the activeness of a user determines their popularity in an environment where many people may know others personally on the network, or if some other cause does (such as friendship). The large clustering towards users with no posts suggests that increased user activity generally does not yield more followers on Campustream. This shows that users follow their friends that they met prior to Campustream and not the users who they find the most interesting based on their posts. The graph below shows the resulting data from March 14th, 2011 to April 13th, 2011.

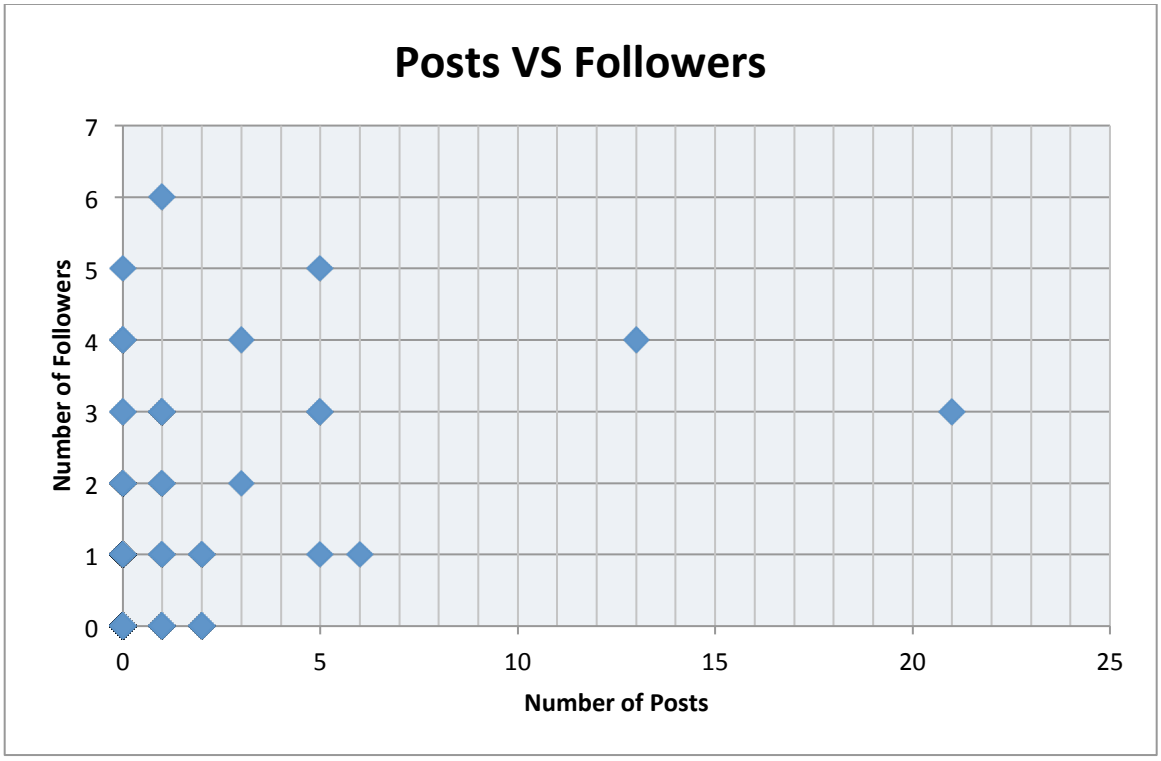


Figure 34: Posts VS Followers

7.14 Number of Followers Distribution

The following graph shows the number of followers of each user. As you can see the majority of users have zero followers, meaning they did not collaborate with friends on the network or look for their friends to follow. It is possible that some users joined prior to their friends and never returned to the website to see if their friends had joined in the future.

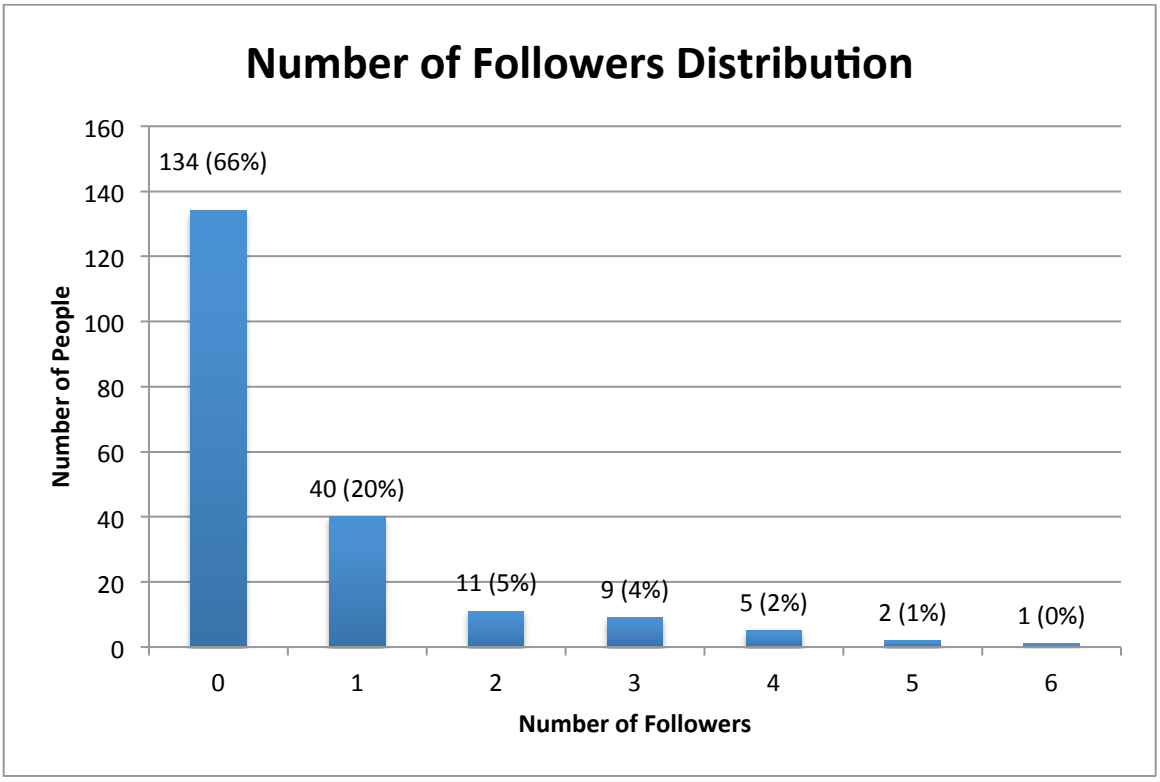


Figure 35: Number of Followers Distribution

7.15 Frequency of Mutual Connections

A mutual connection is when a user if following a person that is following them. This can be used to tell if people are simply following their friends or other users that are following them instead of looking for interesting content posted by those users. As you can see, the majority of users have no mutual connections. This also includes users with no followers, which also meant that they would not have any mutual connections.

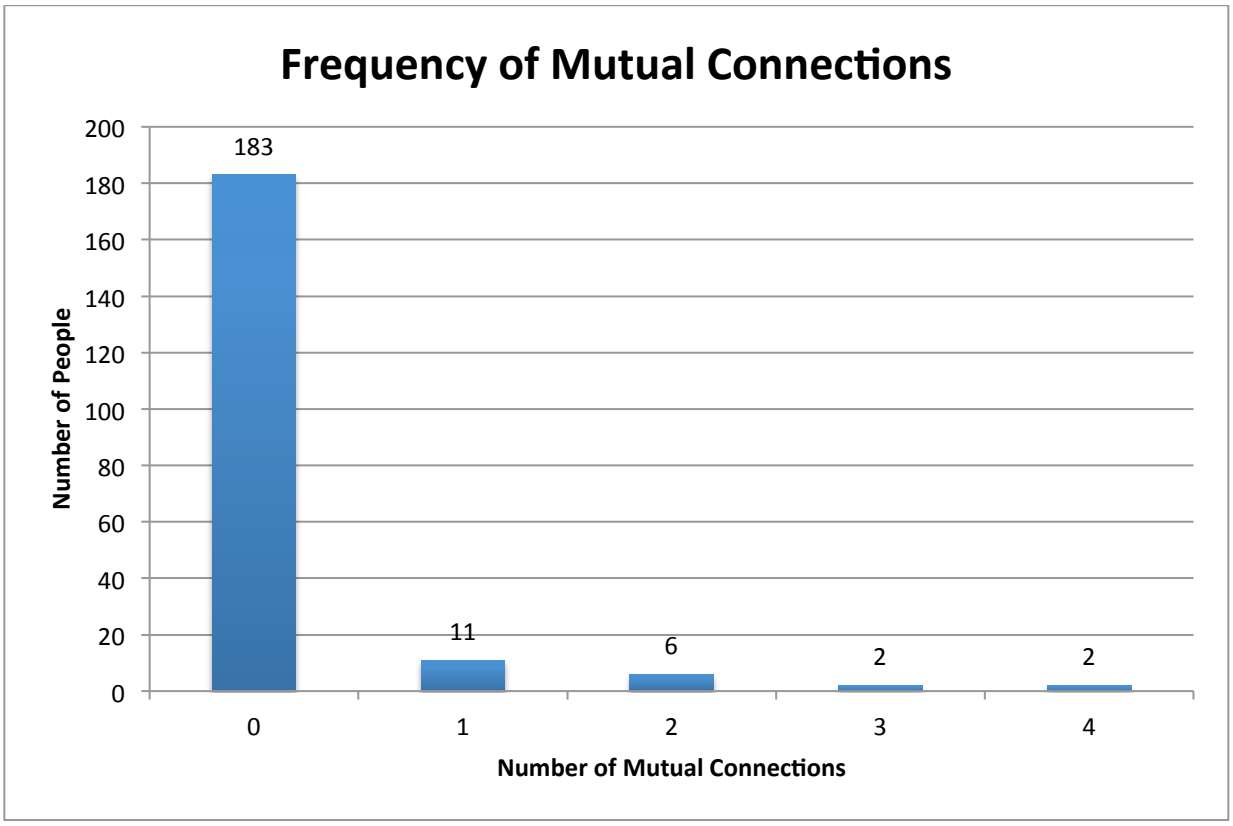


Figure 36: Frequency of Mutual Connections

8 Conclusions

This project studied social interactions of WPI students in an academic social network focused on collaboration and user-submitted content. We studied both the marketing aspect of social networks and the social patterns that emerge over time. The interest shown in Campustream, and the data we were able to gather, has helped us reach some interesting conclusions regarding small social networks and academically based networks.

There is still a lot of potential for Campustream, and a lot of future projects will have the opportunity to expand the site to fit the needs of WPI students even more; however, the core features that have been implemented were more than enough to be able to deduce the conclusions we reached. Many of the students who used Campustream expressed their satisfaction and enjoyment with the service by posting status updates, such as: "This is well done, I really like the layout." and "Awesome site! Nice job!". Posts like this seem to indicate a want for the continual development and improvement of the social network.

9 Future Work

Today's social networks have endless possibilities and countless features. They allow users to upload pictures and videos, are location aware, and they even have advanced facial recognition technologies built right in. Due to the incredible scope of creating a social network from scratch there simply wasn't enough time to add all the features we wanted and because of that there are many ideas we have that could be explored by dozens of future MQPs.

1. **Integration with WPI services**

We had planned to have the network automatically detect which classes each user was taking, or at the very least allow users to enter their class information. This would allow groups to be created within classes, letting users know who else is in their classes as well as who has taken a specific class before. Users could then direct questions at people who have taken a class before when they need information, or easily find someone to work with on a group project.

2. **Location Aware Check-Ins**

Social networks like Foursquare and Facebook allow users to check-in at locations to let other users know they were actually at an event or a restaurant. This could easily be adapted to Campustream as each event already has location information linked to it. By adding GPS coordinates to each location and using Google's Maps APIs it is possible to allow users to check-in at events on campus. This would open the door to a number of features including rewards for attending events, the ability to see who else attended or is attending an event, and even allow the user to see where the event is located on a map.

3. **Event Recommendations**

By knowing which events users have attended in the past, events could automatically be recommended to them in the future. This can be done by looking for key-words in events, seeing who posted the event, and possibly even comparing the times and locations. This would be a great way to help users find new events that they haven't previously heard of and would significantly boost event participation on-campus.

4. **Event Calendar Interface**

A calendar interface for events would automatically place all events on a calendar that can be viewed by users. This calendar could be exportable into outlook or it could even be automatically

linked to users WPI calendars if the services of WPI were more tightly integrated. This would allow users with free time on a specific day to quickly check what events are occurring on that day and see what they would like to attend.

5. **Entire Site Search Features**

Currently users can only search specific sections of the website. For instance, this allows them to search for tags in the Collaboration section as well as users by their names. However, there is no overall search bar that allows users to search the entire website for something. This would be a simply addition to greatly improve the usability of the website.

6. **Automatic Importing of News and Events**

Originally we had planned to automatically import news from the WPI Connection and events from Facebook. This can be done by using the RSS feed of the WPI Connection to quickly search for new news articles and can be done on Facebook by looking at the events of users with their accounts linked and seeing if they are relevant to WPI and open to the public. One of the biggest problems with the site is the lack of content and this would be a great way to alleviate that issue.

7. **On-Site Private Messaging**

Currently users can email each other by clicking on their usernames within their profiles. This allows direct contact but it is far from Facebook's private messaging that allows users to see who is online and message them instantly. On-Site private messaging would allow users to quickly see who is online and chat with them about anything. This would also increase the amount of time users spend on the site if they stay there to chat.

8. **Direct Photo and Video Uploads**

Currently users are able to upload photos or videos to other hosting services and link to them on Campustream. Some content items are displayed directly inline, but going to an external source to upload your content is a hassle. Allowing users to directly upload photos and videos was a major part of this project; unfortunately bandwidth and speed concerns forced us to eliminate those features. This is something that would drastically improve the site and is certainly one of the first new features that should be added in the future.

References

[1] Android Open Source Project. (2010, September 1). *Platform Versions*. Retrieved September 15, 2010, from Android Developers: http://developer.android.com/resources/dashboard/platform-versions.html

[2] Android Open Source Project. (2010, September 12). *Tools Overview*. Retrieved September 15, 2010, from Android Developers: http://developer.android.com/guide/developing/tools/index.html

[3] Burns, B. (n.d.). *DroidDraw: Graphical User Interface Editor for Android Cell Phone Development and Programming*. Retrieved September 15, 2010, from DroidDraw: http://www.droiddraw.org/

[4] Claburn, T. (2010, July 12). *Google App Inventor Simplifies Android Programming*. Retrieved September 15, 2010, from InformationWeek SMB: http://www.informationweek.com/news/smb/mobile/showArticle.jhtml?articleID=225702880&subSection=News

[5] Douglas, N. (2007 йил 12-March). *Twitter blows up at SXSW Conference*. Retrieved 2010 йил 8-September from http://gawker.com/tech/next-big-thing/twitter-blows-up-at-sxsw-conference-243634.php

[6] Facebook. (2010 йил August). *Company Timeline*. Retrieved 2010 йил 8-September from http://www.facebook.com/press/info.php?timeline

[7] Facebook. (2010). *Facebook for Android*. Retrieved September 15, 2010, from Facebook: http://www.facebook.com/apps/application.php?id=74769995908&v=info

[8] Facebook. (2010). *Facebook for Blackberry smartphones*. Retrieved September 15, 2010, from Facebook: http://www.facebook.com/apps/application.php?id=2254487659&v=info

[9] Facebook. (2010). *Facebook for iPhone*. Retrieved September 15, 2010, from Facebook: http://www.facebook.com/iphone?v=info

[10] gseth. (2010, August 7). *Smartphone Market Share in United States*. Retrieved September 15, 2010, from eSeth: http://eseth.net/blog/87/smartphone-market-share-in-united-states

[11] Gurp, J. v. (2008, August 19). *Google Android: Initial Impressions and Criticism*. Retrieved September 15, 2010, from javalobby: http://www.javalobby.org/nl/archive/jlnews_20071113o.html

[12] Higginbotham, S. (2010, September 8). *Digg Not Likely to Give Up on Cassandra*. Retrieved October 7, 2010, from Gigaom: http://gigaom.com/2010/09/08/digg-not-likely-to-give-up-on-cassandra/

[13] Ho, A., Maiga, A., & Aimeur, E. (2009). *Privacy Protection Issues in Social Networking Sites.* Montreal: IEEE.

[14] Hollister, S. (2010, September 4). *Android accounts for one-quarter of mobile web, says Quantcast*. Retrieved September 15, 2010, from Engadget: http://www.engadget.com/2010/09/04/android-accounts-for-one-quarter-of-mobile-web-traffic-says-qua/

[15] Jamali, S., & Rangwala, H. (2009). *Digging Digg: Comment Mining, Popularity Prediction, and Social Network Analysis.* IEEE.

[16] Miller, C. C. (2010 йил 18-June). *Sports Fans Break Records on Twitter*. Retrieved 2010 йил 8-September from http://bits.blogs.nytimes.com/2010/06/18/sports-fans-break-records-on-twitter/

[17] Morris, M. R., Teevan, J., & Panovich, K. (2010). *What Do People Ask Their Social Networks, and Why? A Survey Study of Status Message Q&A Behavior.* Atlanta: ACM.

[18] Nickson, C. (2009 йил 21-January). *The History of Social Networking*. Retrieved 2010 йил 8-September from http://www.digitaltrends.com/features/the-history-of-social-networking/

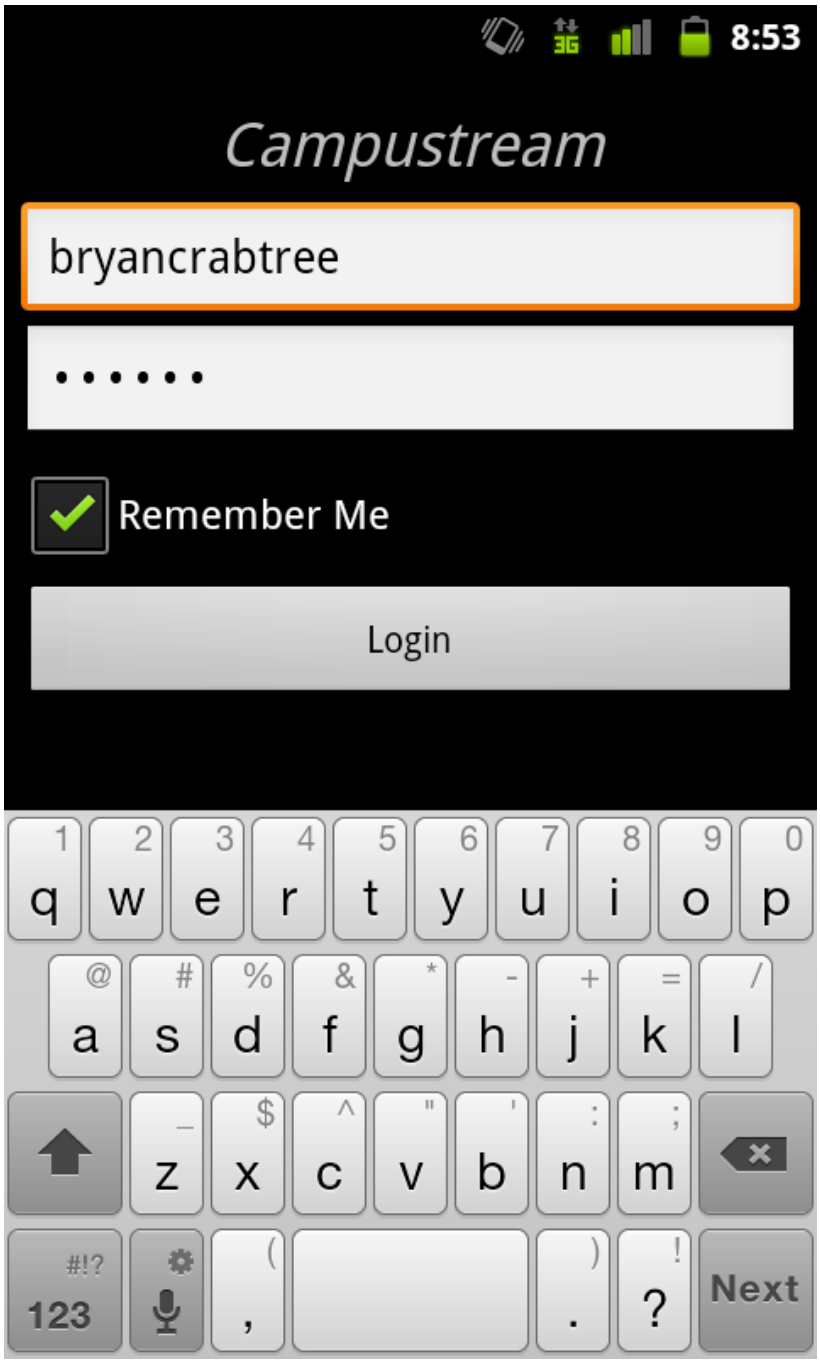
[19] Popescu, A. (2010, February 10). *Geolocation API Specification*. Retrieved September 15, 2010, from W3C: http://dev.w3.org/geo/api/spec-source.html

[20] The Nielsen Company. (2009 йил 9-March). *Social Networking's New Global Footprint*. Retrieved 2010 йил 6-9 from Nielsen Wire: http://blog.nielsen.com/nielsenwire/nielsen-news/social-networking-new-global-footprint/

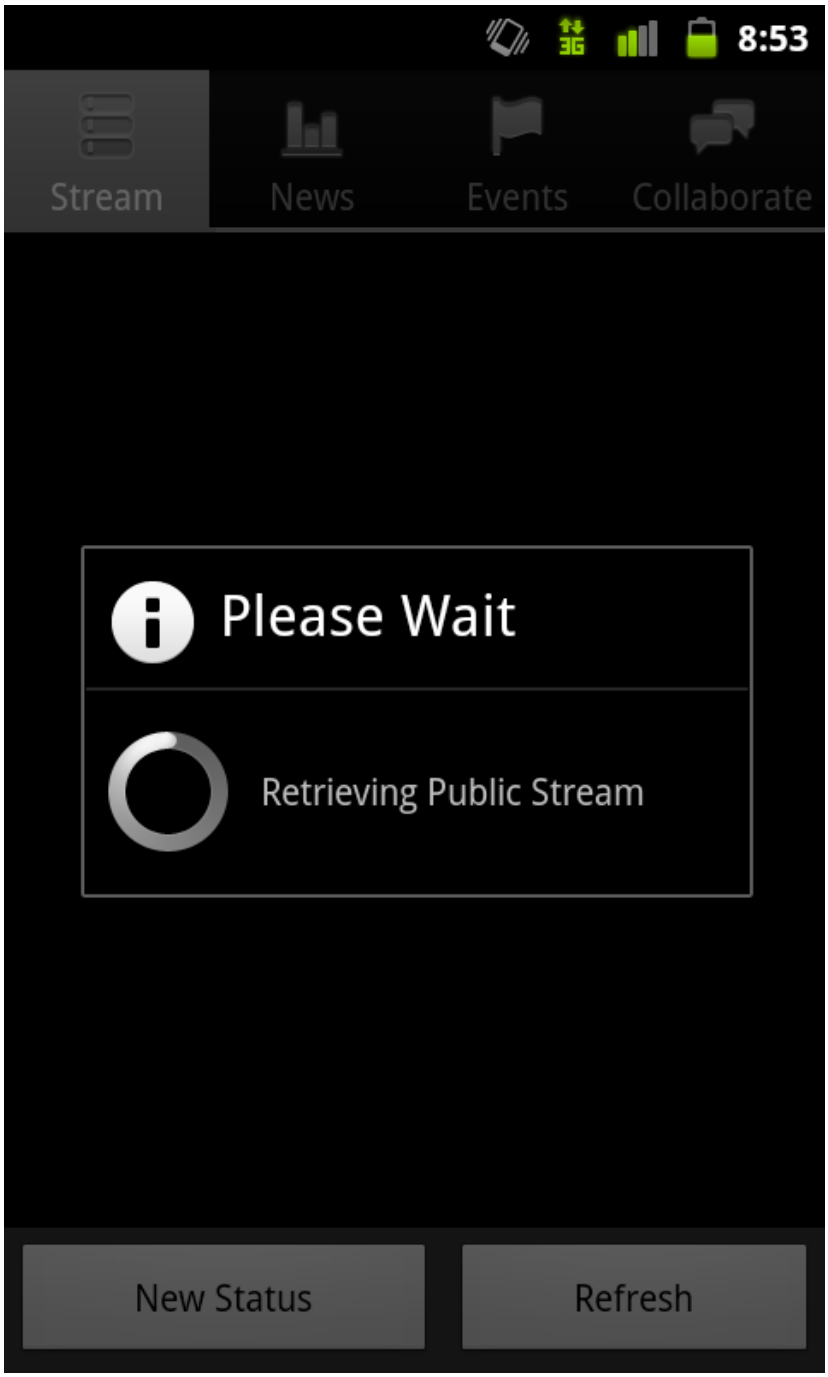
[21] Wikipedia. (2010, October 6). *MySQL*. Retrieved October 6, 2010, from Wikipedia: http://en.wikipedia.org/wiki/Mysql

Appendix A – Android Screenshots

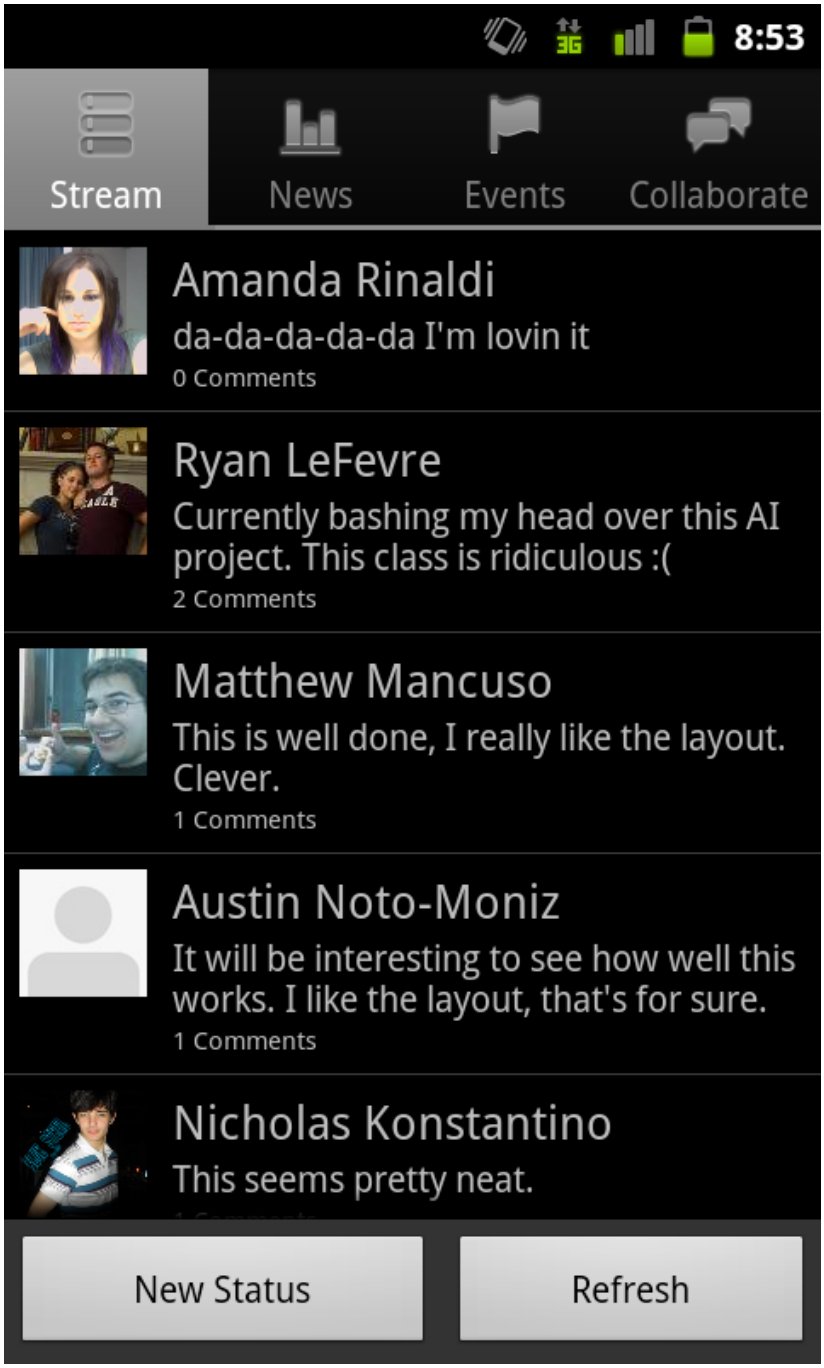
Login Screen



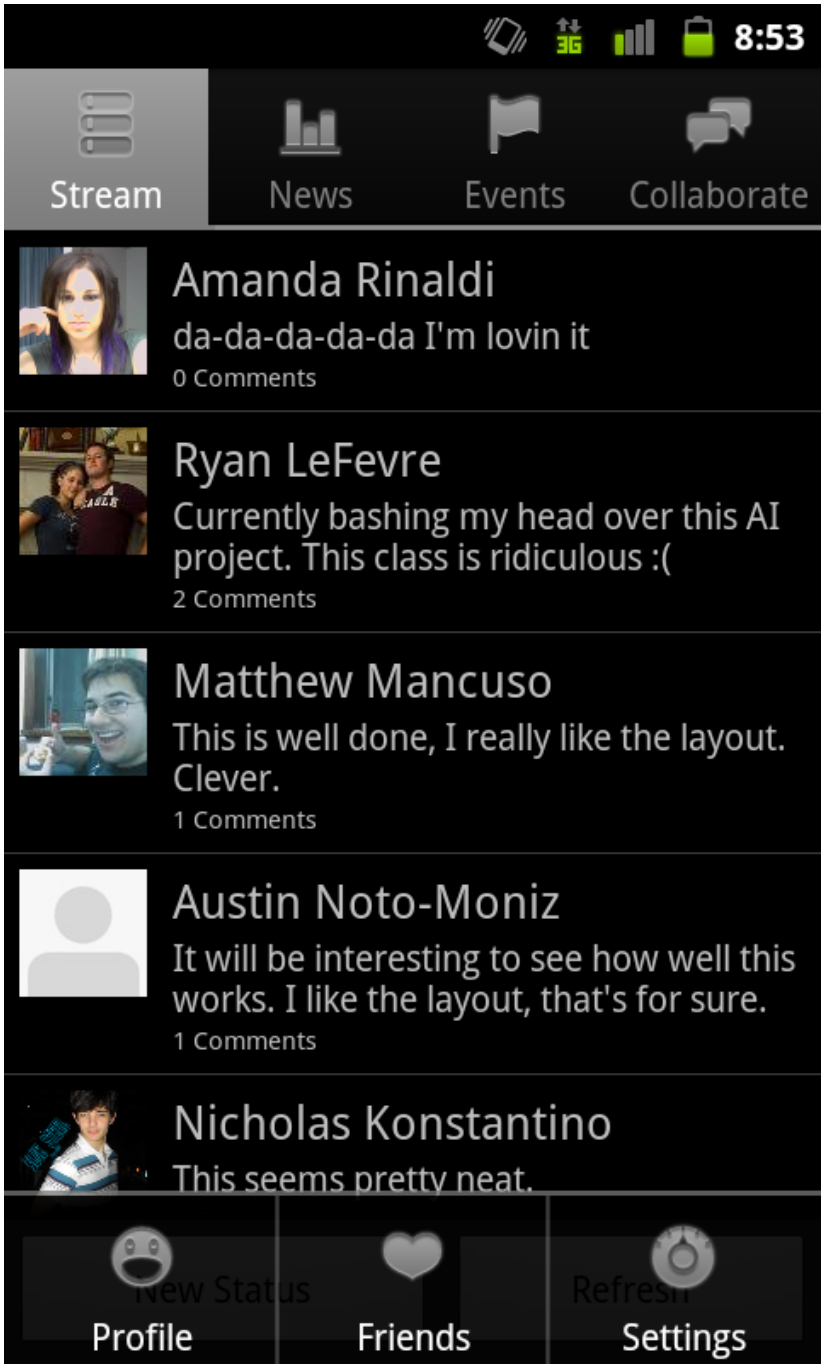
Loading Dialog



Stream Tab


8:53
Stream
News
Events
Collaborate
Amanda Rinaldi
da-da-da-da-da I'm lovin it
0 Comments
Ryan LeFevre
Currently bashing my head over this AI project. This class is ridiculous :(
2 Comments
Matthew Mancuso
This is well done, I really like the layout. Clever.
1 Comments
Austin Noto-Moniz
It will be interesting to see how well this works. I like the layout, that's for sure.
1 Comments
Nicholas Konstantino
This seems pretty neat.
New Status
Refresh

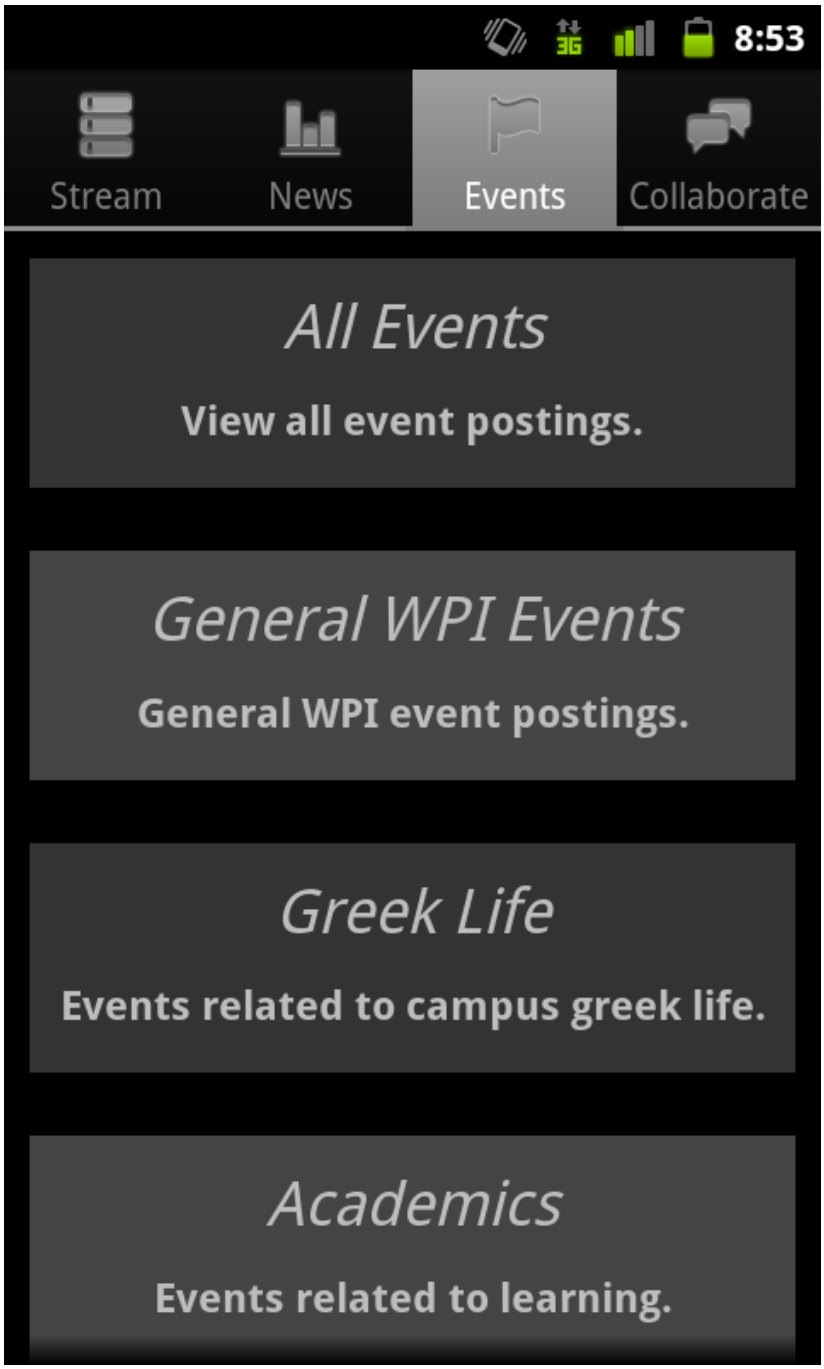
Stream Tab w/ Options



News Tab

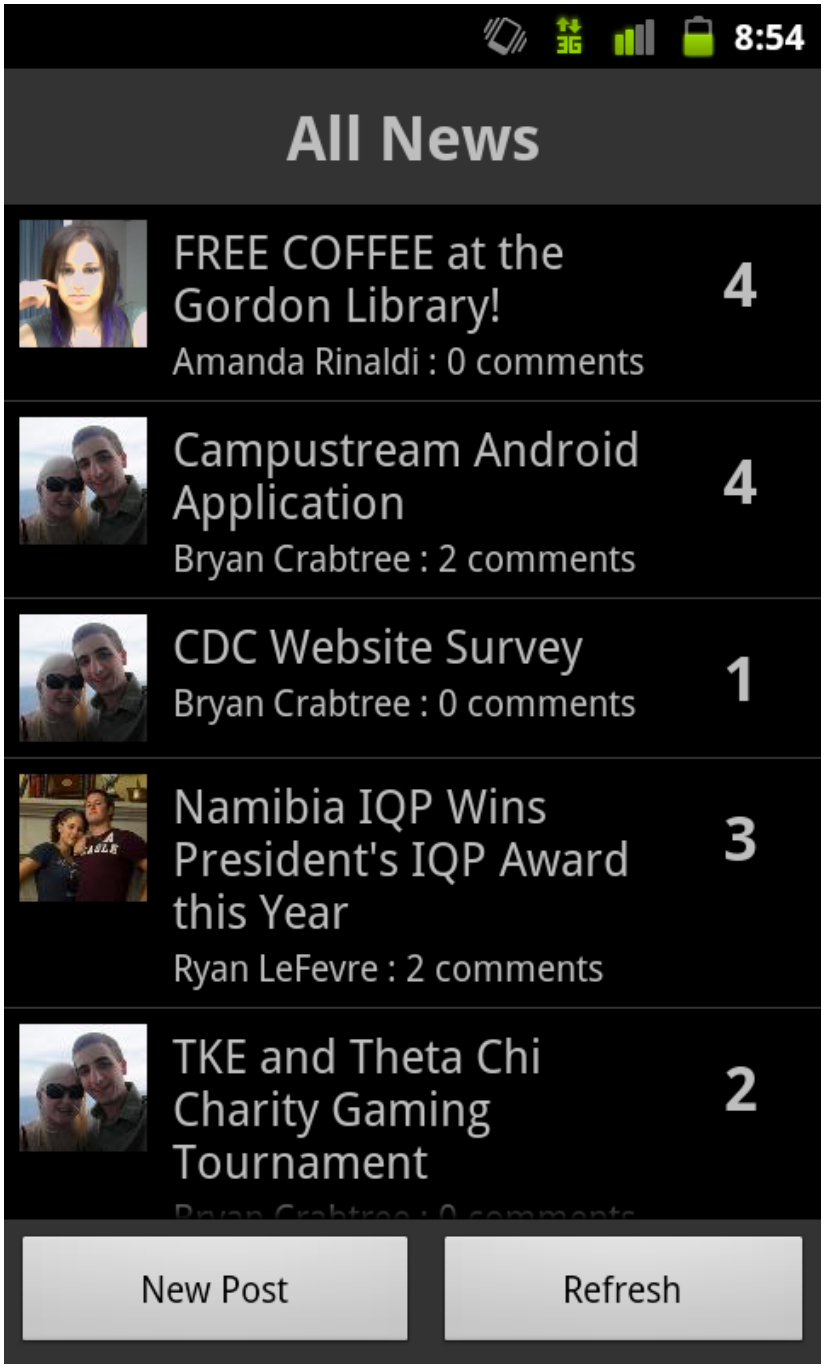

8:53
Stream
News
Events
Collaborate
All News
View all news postings.
General WPI News
News about WPI in general.
Greek Life
News related to campus greek life.
Academics
Class and lecture news.

Events Tab

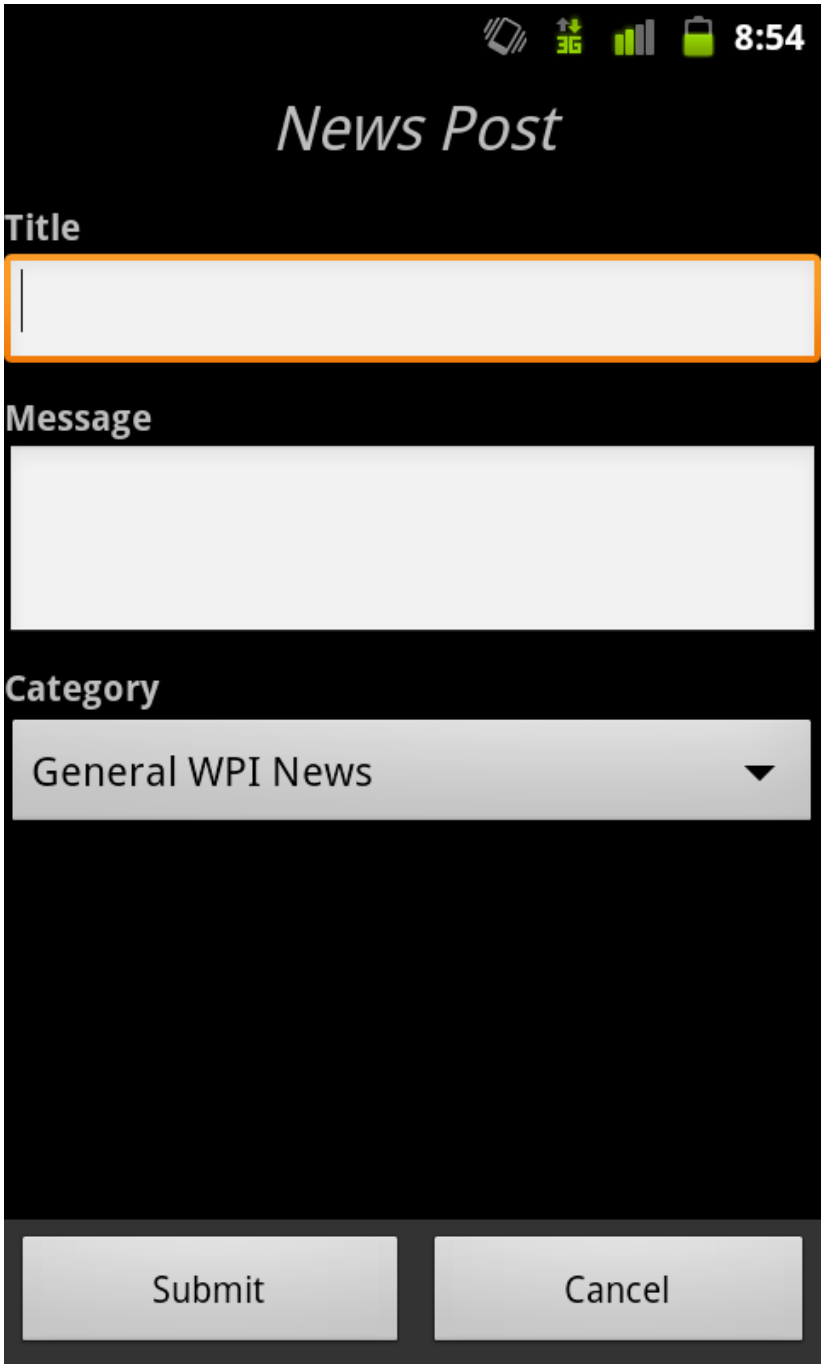

8:53
Stream
News
Events
Collaborate
All Events
View all event postings.
General WPI Events
General WPI event postings.
Greek Life
Events related to campus greek life.
Academics
Events related to learning.

Collaboration Tab

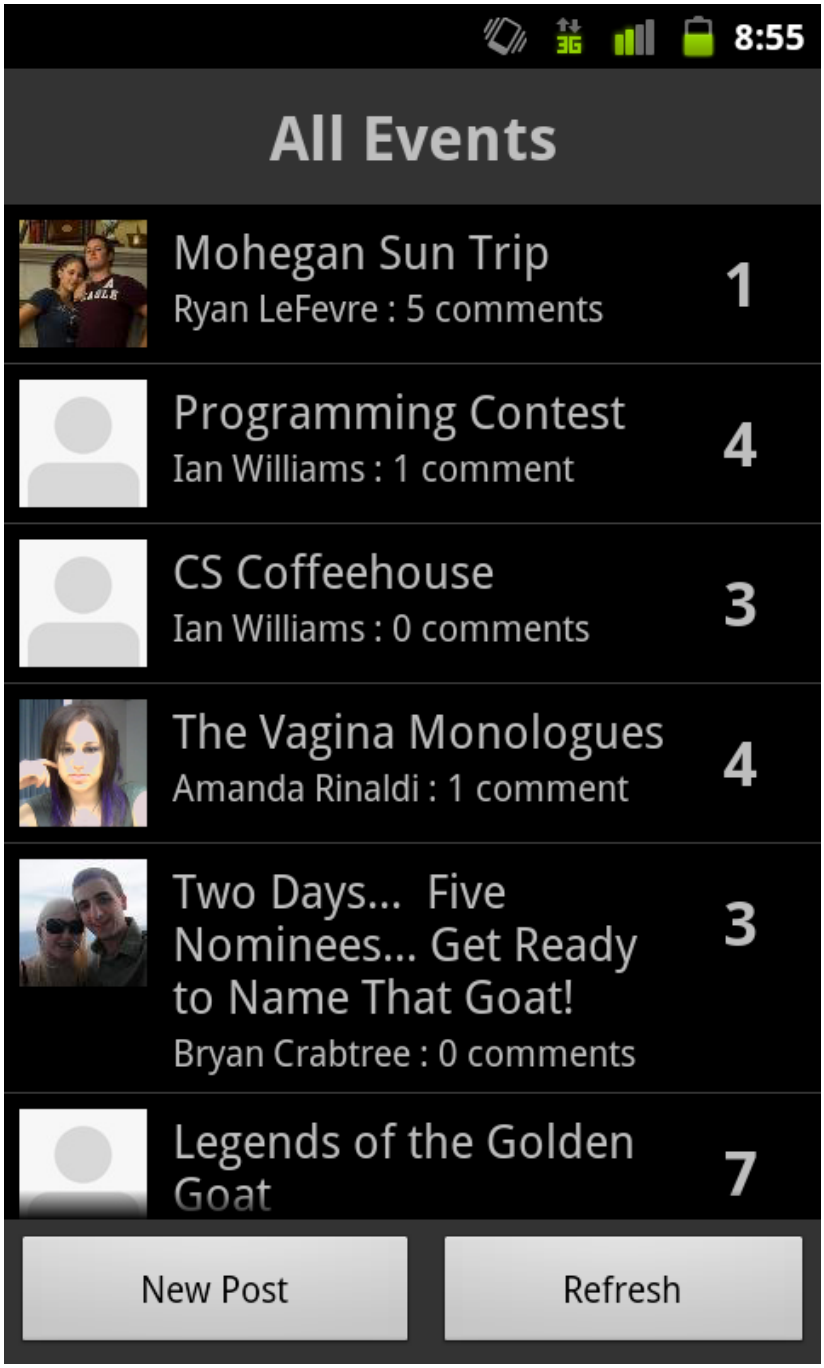
Expanded News View



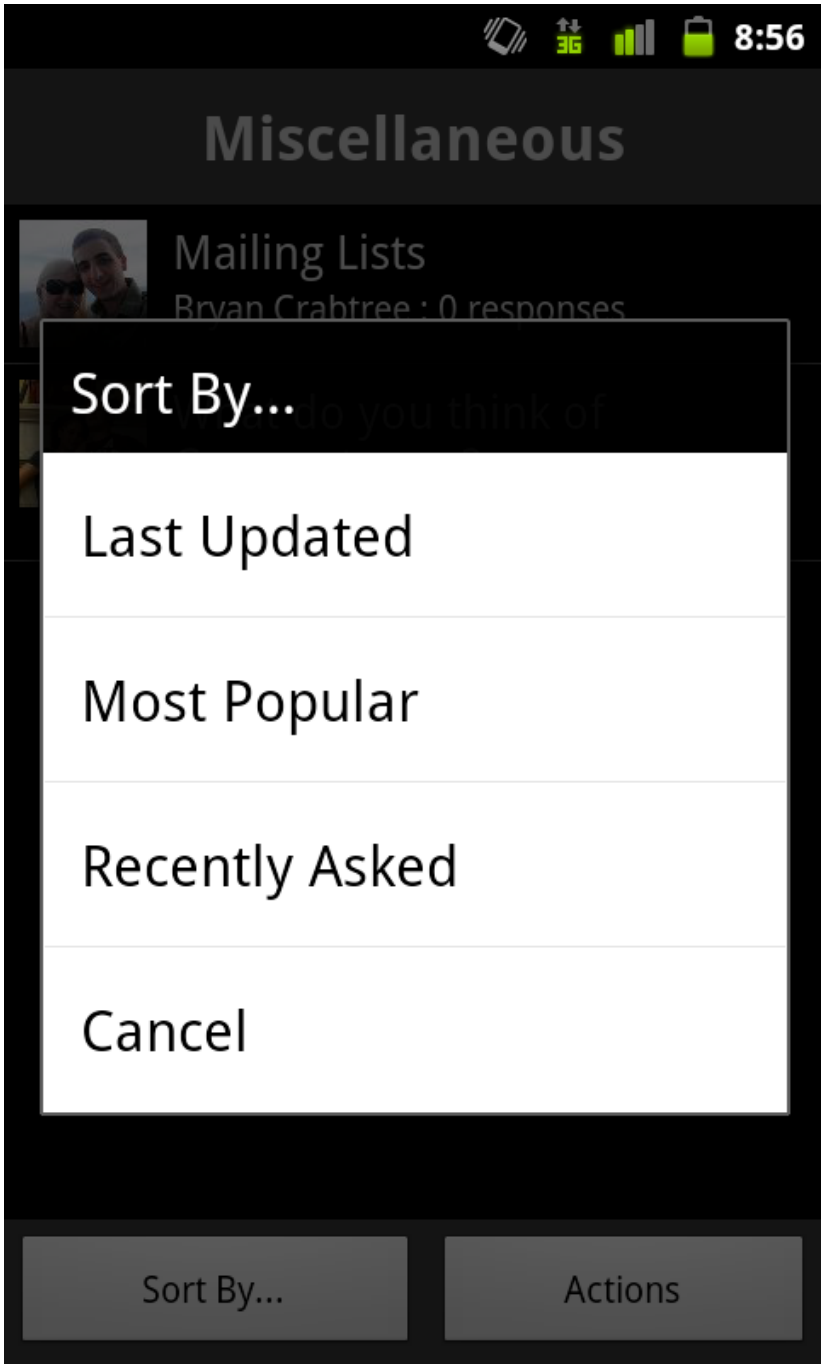
News Post Dialog



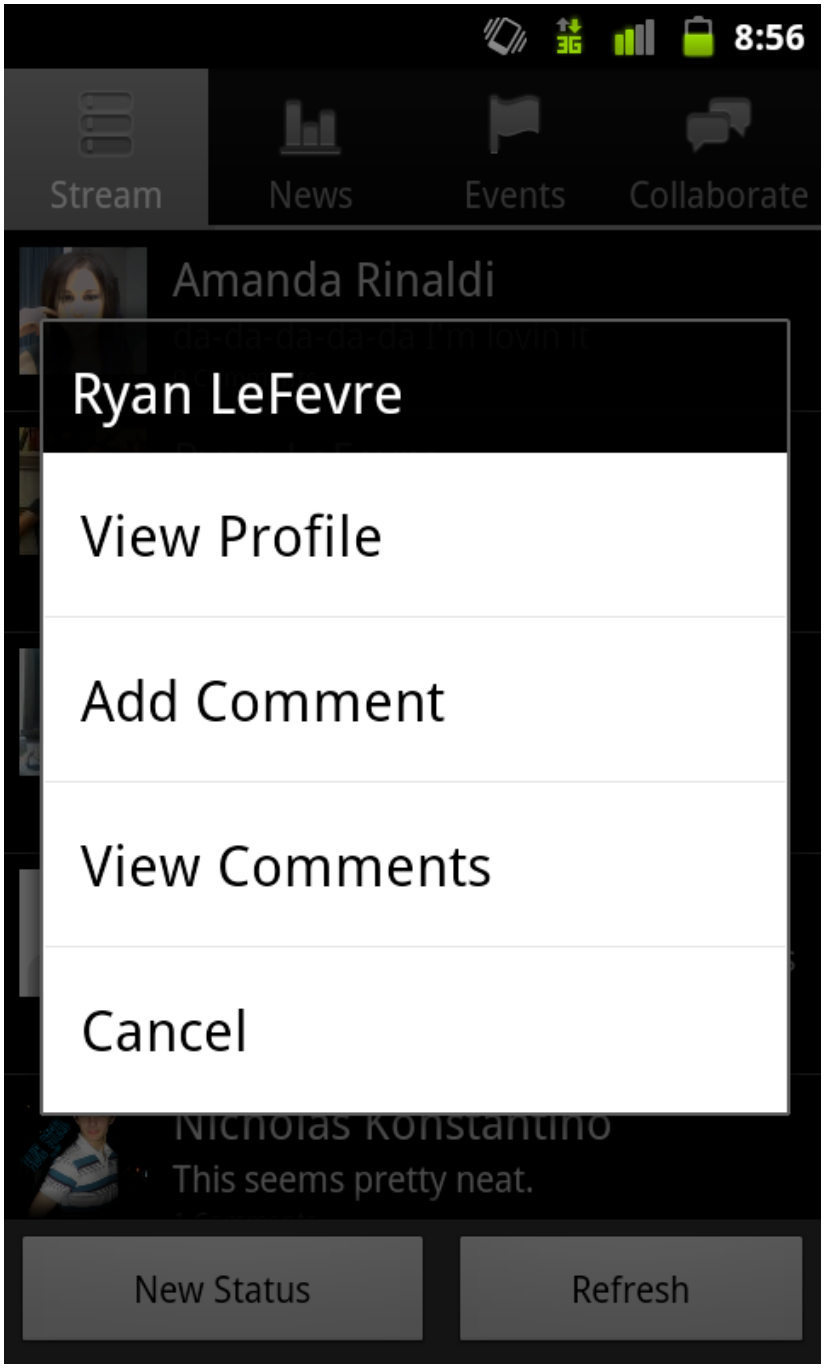
Expanded Events View



Sorting Dialog

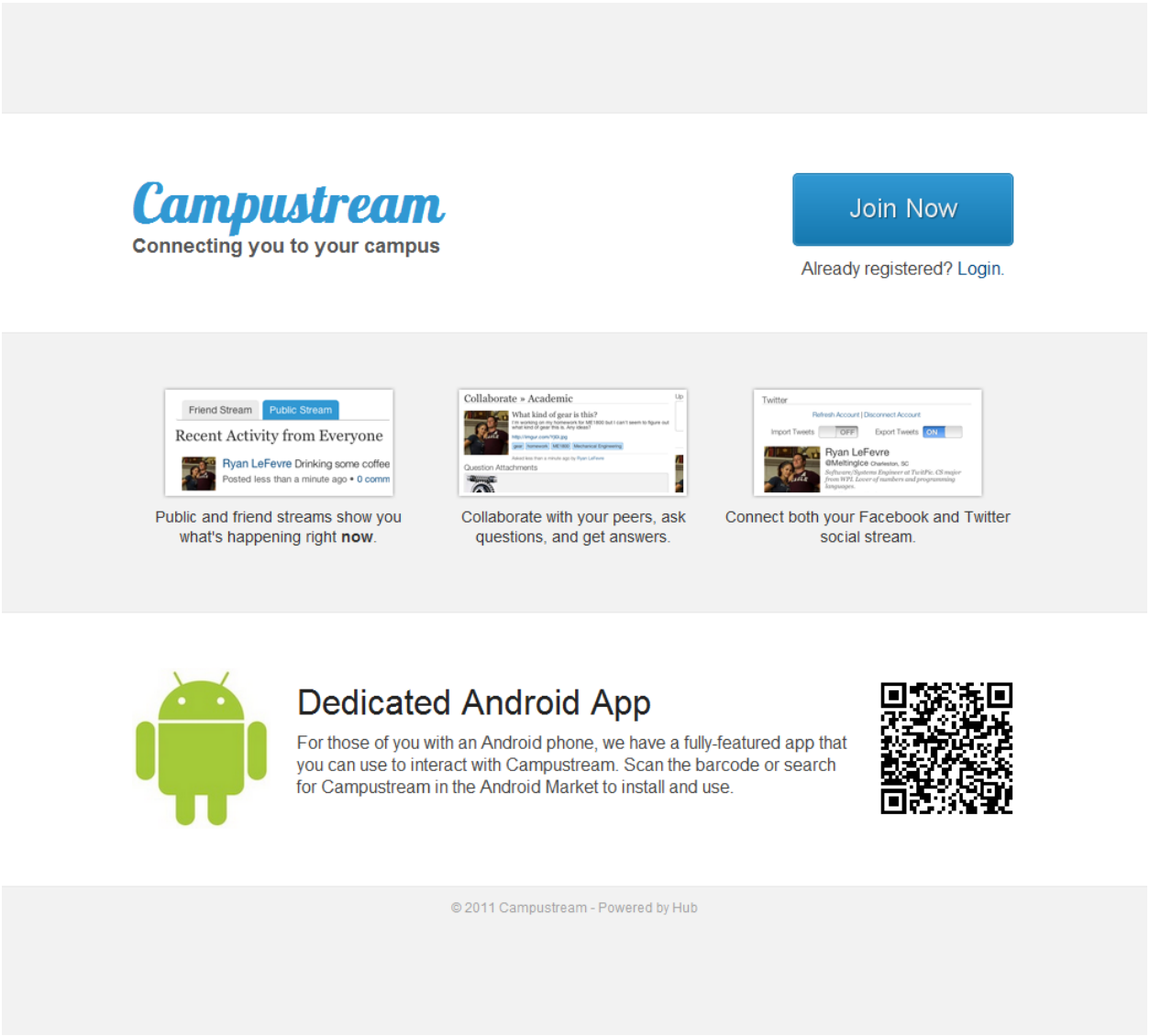


Stream Popup Dialog

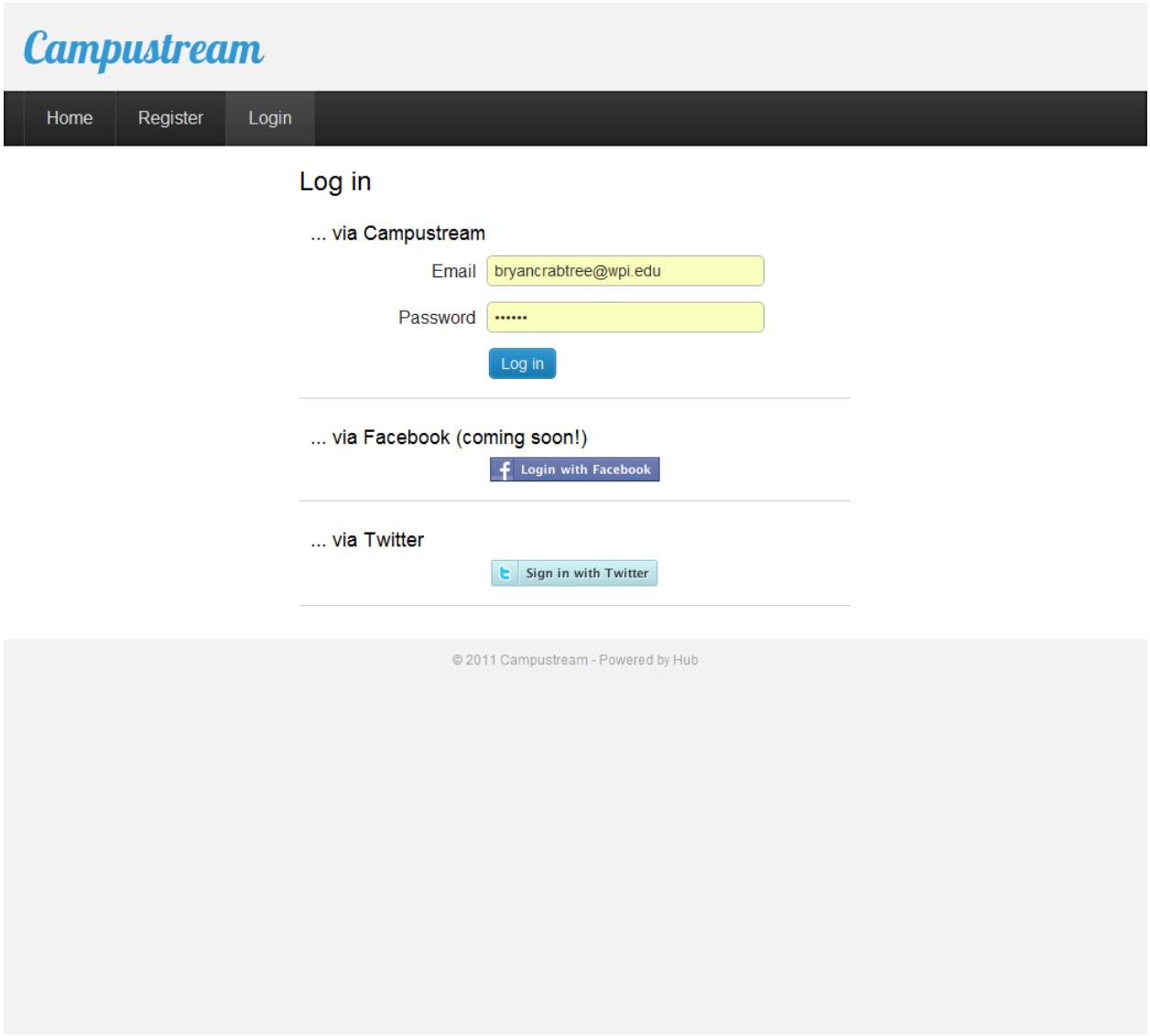


Appendix B – Website Screenshots

Main Page



Login


Campustream
Home
Register
Login
Log in
... via Campustream
Email
bryancrabtree@wpi.edu
Password
Log in
... via Facebook (coming soon!)
Login with Facebook
... via Twitter
Sign in with Twitter
© 2011 Campustream - Powered by Hub

Stream

Campustream

Home | Profile | News & Events | Collaborate | Users | Settings | Logout

Friend Stream | Public Stream

Recent Activity from Everyone

Rebecca Meissen replied to a question and said, "I think you're on to something really interesting here! I think Boyd's characteristics (persistence, reliability, searchability, etc.) set the stage for what I like to think of as a massive scavenger hunt. By word of mouth this video went viral, and like all things viral people just HAD to watch it (no matter how horrible their friends told them it was)!

It's not just a song, and it's not just a video-- it's an experience! The shared experience (temporal/spatial cult deWinter talked about) of finding the video and sitting through the singing. In this way, it's no wonder we see so many more unpleasant videos circulating rapidly through networks than pleasant videos. This has to do with the type of material friends are willing to talk about, etc. Usually they exclaim about the horrible, rather than praise the wonderful.

Why isn't there an equally famous, but well-acclaimed video out there? And if there is, why haven't we heard about it?"

Replied about 23 hours ago • 0 comments

Natasha Bonina What are karma points?

Posted 2 days ago • 0 comments

Natasha Bonina replied to a question and said, "This is an interesting project. I have only been personally exposed to the song once (in music video form) and immediately was turned off by her lack of talent in the vocal department as well as the bland lyrics. With just radio, she probably wouldn't have received half as much publicity."

Replied 3 days ago • 0 comments

Ryan LeFevre Taking a shower while an automated script analyzes the Campustream social graph. Such is the life of a programmer.

Posted 3 days ago • 0 comments

Natasha Bonina The color blue seems to be popular for social networking sites... FB, Twitter, and now Campustream. Hmmm

Posted 4 days ago • 1 comments

Update your status

Broadcast to: Everyone Followers Only Post

Bryan Crabtree

bryancrabtree

Campustream co-founder.

http://www.campustream.com

Class of 2011

Major(s): Computer Science

Minors(s): None

24 Karma	5 Following	4 Followers

0

Recent Activity

Campustream

Home | Profile | News & Events | Collaborate | Users | Settings | Logout

Recent Activity

Recent Activity from Ryan LeFevre

Ryan LeFevre Taking a shower while an automated script analyzes the Campustream social graph. Such is the life of a programmer.
Posted 3 days ago • 0 comments

Ryan LeFevre Campustream contest: http://campustream.com/ne/l
Posted 4 days ago • 0 comments

Ryan LeFevre replied to a question and said, "Sounds awesome, good luck!"
Replied 4 days ago • 0 comments

Ryan LeFevre Today's grocery list: coffee, coffee creamer, and deodorant. Pretty much everything a programmer needs.
Posted 9 days ago • 1 comments

Ryan LeFevre Wow the weather today is awesome! Too bad I'll probably be stuck inside working on homework :(
Posted 10 days ago • 0 comments

Ryan LeFevre Thanks to everyone for the great feedback so far! We've definitely getting a good idea of what the site needs and what is working well. All of this will go into our MQP analysis and hopefully future years will be up to the task :)
Posted 11 days ago • 0 comments

Ryan LeFevre replied to a question and said, "Thanks for the feedback so far everyone! I added links to user's profiles on this page, can't believe I forgot that one.

I agree that it would be nice to be able to integrate classes and groups, and I would love for future MQP's to be able to add this functionality. We simply didn't have time this year to add everything we wanted!"
Replied 11 days ago • 0 comments

Ryan LeFevre replied to a question and said, "I heard about this today, and

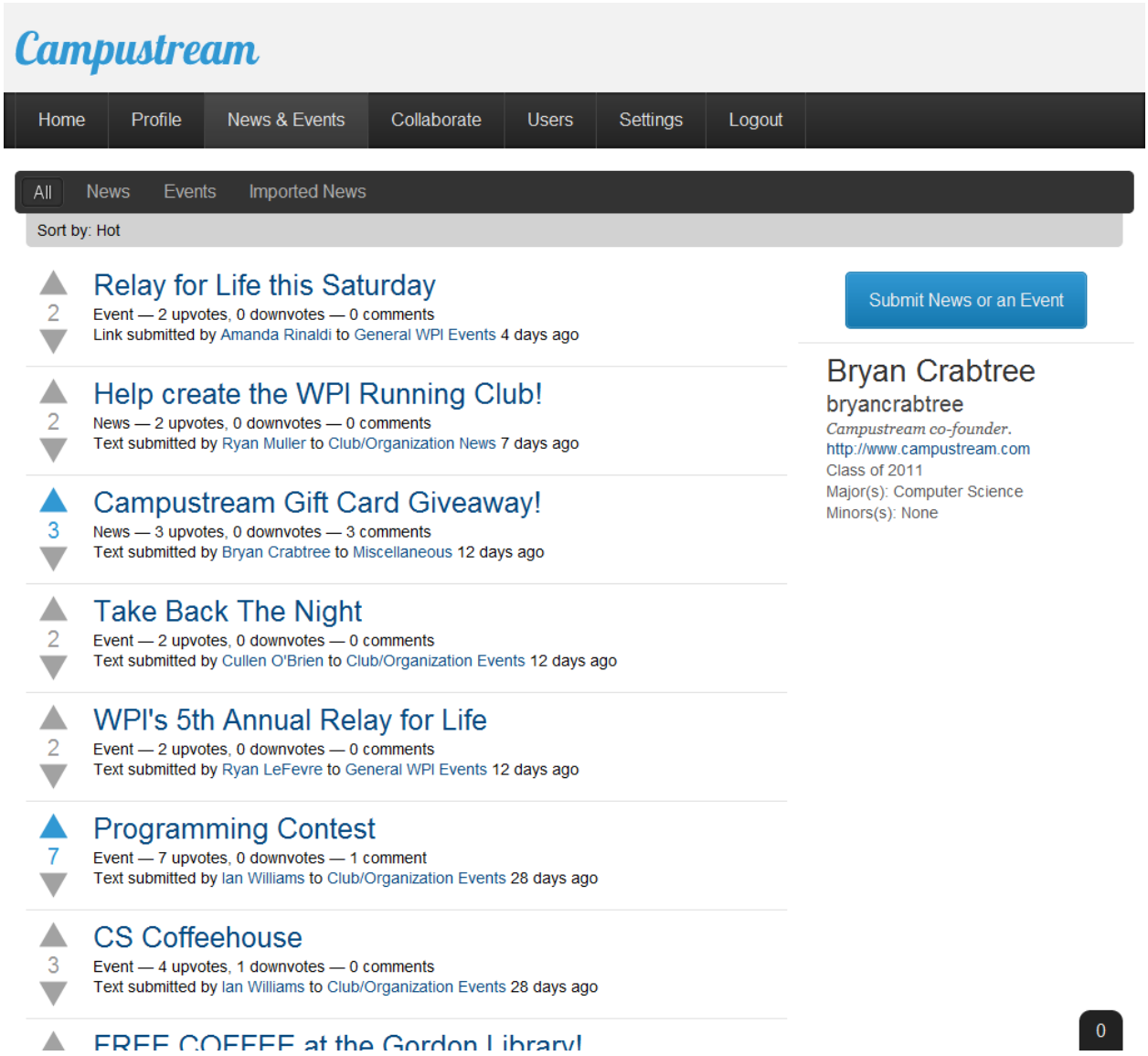
Ryan LeFevre
rlefevre
Co-founder of Campustream and Software/Systems Engineer at TwitPic
http://meltingice.net
Class of 2011
Major(s): Computer Science

Following

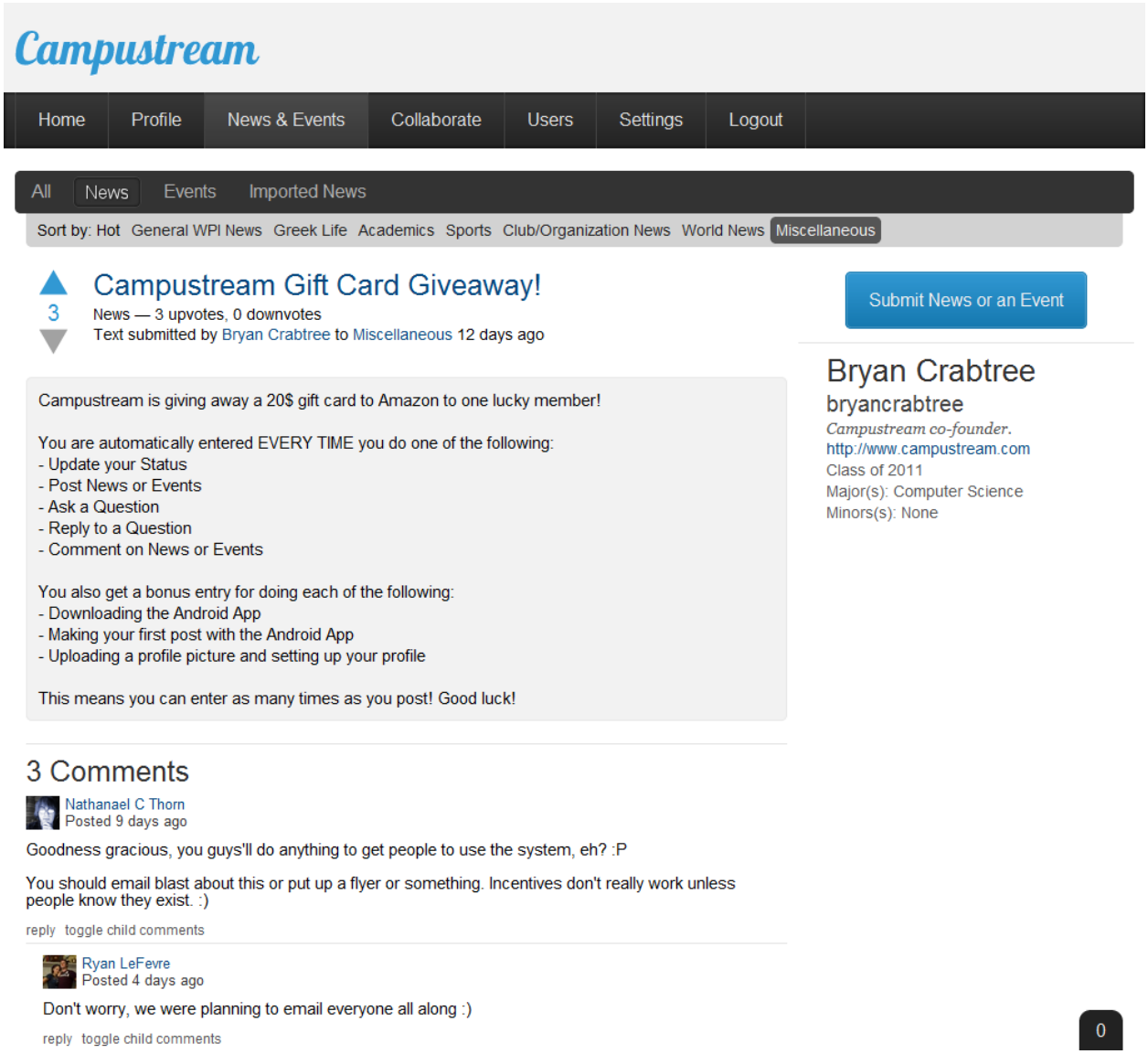
18 Karma	12 Following	6 Followers

0

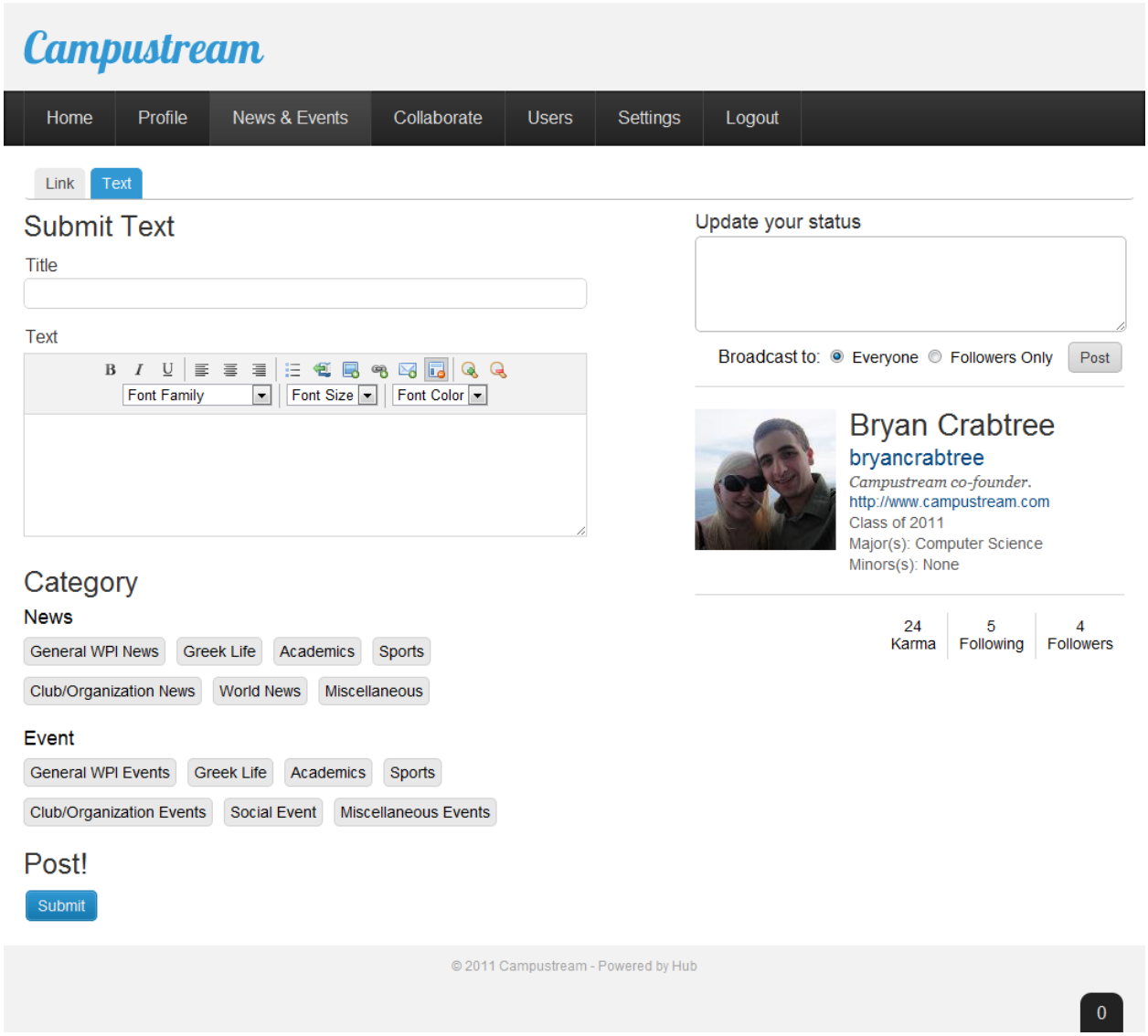
News and Events



News Post


Campustream
Home
Profile
News & Events
Collaborate
Users
Settings
Logout
All
News
Events
Imported News
Sort by: Hot
General WPI News
Greek Life
Academics
Sports
Club/Organization News
World News
Miscellaneous
Campustream Gift Card Giveaway!
3
News — 3 upvotes, 0 downvotes
Text submitted by Bryan Crabtree to Miscellaneous 12 days ago
Campustream is giving away a 20$ gift card to Amazon to one lucky member!
You are automatically entered EVERY TIME you do one of the following:
- Update your Status
- Post News or Events
- Ask a Question
- Reply to a Question
- Comment on News or Events
You also get a bonus entry for doing each of the following:
- Downloading the Android App
- Making your first post with the Android App
- Uploading a profile picture and setting up your profile
This means you can enter as many times as you post! Good luck!
3 Comments
Nathanael C Thorn
Posted 9 days ago
Goodness gracious, you guys'll do anything to get people to use the system, eh? :P
You should email blast about this or put up a flyer or something. Incentives don't really work unless people know they exist. :)
reply toggle child comments
Ryan LeFevre
Posted 4 days ago
Don't worry, we were planning to email everyone all along :)
reply toggle child comments
Submit News or an Event
Bryan Crabtree
bryancrabtree
Campustream co-founder.
http://www.campustream.com
Class of 2011
Major(s): Computer Science
Minors(s): None
0

Event Submission



Collaboration

Campustream

Home | Profile | News & Events | Collaborate | Users | Settings | Logout

Collaborate

Please note that we do not condone or allow cheating for classes in any way, shape, or form. If you post content on Campustream that is considered cheating, then you are responsible for it and it may be reported to WPI.

Classes and Projects
Have a question about some homework or a school project? Anything academic-related can be asked here.

Social
Want to know what's going on this weekend? Maybe you want to carpool to a movie? Any social questions can be asked here.

Miscellaneous
If there's something on your mind that you need to ask, but doesn't fit in with academics or the social-scene, then this is the place to ask it.

Campus Happenings
Work together to let other students know what's happening around campus!

Update your status

Broadcast to: Everyone Followers Only Post

Bryan Crabtree
bryancrabtree
Campustream co-founder.
http://www.campustream.com
Class of 2011
Major(s): Computer Science
Minors(s): None

24 Karma	5 Following	4 Followers

© 2011 Campustream - Powered by Hub

0

Friend List

Campustream

Home | Profile | News & Events | Collaborate | Users | Settings | Logout

Users Bryan Crabtree follows

Ryan LeFevre rlefevre

Co-founder of Campustream and Software/Systems Engineer at TwitPic

Following

Justin Stocker jstocker

I'm a deer.

Following

Adrian Catarius acatarius

Following

Lucas Scotta lscotta

Following

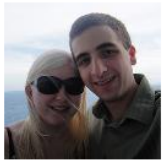
Ian Williams ianw

Ex-president of the ACM. Sigma Pi & Ultimate Frisbee player. Software Engineer at Microsoft, starting in July 2011.

Following

Update your status

Broadcast to: Everyone Followers Only Post



Bryan Crabtree

bryancrabtree

Campustream co-founder.

http://www.campustream.com

Class of 2011

Major(s): Computer Science

Minors(s): None

24 Karma	5 Following	4 Followers

0

Appendix C – Campustream APIs

API format: http://api.campustream.com/1/ + API endpoint + .(Json | xml)

User Endpoints

user/authorize

Retrieve the access token for the specified user.

- Type: POST
- Formats: Plain Text, JSON
- Parameters:
    - username (required)
    - password (required, SHA1 hashed)

users/show

Retrieve information about the given user.

- Type: GET or POST
- Authentication: false
- Formats: JSON, XML
- Parameters:
    - username (required)

Status Endpoints

statuses/public_timeline

Retrieve the last (up to) 200 statuses from the public timeline.

- Type: GET
- Authentication: false
- Formats: JSON
- Parameters:
    - count (max # of statuses, 1 - 199, optional)

statuses/user_timeline

Retrieve the last (up to) 200 statuses for the specified user.

- Type: GET
- Authentication: false
- Formats: JSON
- Parameters:
    - username (required)
    - count (max # of statuses, 1 - 199, optional)

statuses/show

Retrieve information about the given status.

- Type: GET
- Authentication: false
- Formats: JSON, XML
- Parameters:
    - id (status id, required)

statuses/update

Posts a new status message for the authenticated user

- Type: POST
- Authentication: true
- Formats: XML, JSON
- Parameters:
    - message (required)
    - type (optional, (text or link for now)

    - public (optional, defaults to true)

statuses/comments

Retrieve all comments for the status posting

- Type: GET
- Authentication: false
- Formats: XML, JSON
- Parameters:
    - status_id (required)

Comment Endpoints

comments/update

Posts a comment for a given status.

- Type: POST
- Authentication: true
- Formats: XML, JSON
- Parameters:
    - status_id (required)
    - message (required)

Friendship Endpoints

friendships/show

Shows the relation between two given users. You must specify source and target user info, can use user ID or username for either.

- Type: GET
- Authentication: false
- Formats: JSON, XML
- Parameters:
    - source_id or source_username (required)
    - target_id or target_username (required)

friendships/create

Makes the authenticated user follow the given user. Returns the followed user.

- Type: POST
- Authentication: true
- Formats: XML, JSON
- Parameters:
    - user_id or username (required)

friendships/destroy

Makes the authenticated user unfollow the given user. Returns the unfollowed user.

- Type: POST
- Authentication: true
- Formats: XML, JSON
- Parameters:
    - user_id or username (required)

friendships/follows

Returns a list of users that the given user follows.

- Type: GET
- Authentication: false
- Formats: JSON
- Parameters:
    - user_id or username (required)

friendships/followers

Returns a list of users that follow the given user.

- Type: GET
- Authentication: false
- Formats: JSON
- Parameters:
    - user_id or username (required)

Collaboration Endpoints

collaborate/list

Lists questions in the current section.

- Type: GET
- Authentication: false
- Formats: JSON, XML
- Parameters:
    - section (academic, social, misc, campuswatch; optional)
    - sorting (last_updated, most_popular, newest, search; optional)
    - page (optional)

collaborate/show

Shows the given question and responses.

- Type: GET
- Authentication: false
- Formats: XML, JSON
- Parameters:
    - Must be one of the following present:
    - short_id (ID shown in URL; optional)
    - id (actual ID of question; optional)

collaborate/create

Creates a new question in the specified section.

- Type: POST
- Authentication: true
- Formats: XML, JSON
- Parameters:
    - title (question title; required)
    - category (question section/category; required)
    - message (question body; required)
    - public (0 = friends only, 1 = public stream; optional)

collaborate/respond

Post a response to a question.

- Type: POST
- Authentication: true
- Formats: XML, JSON
- Parameters:
    - Must have short_id or id
    - short_id (question short_id; optional)
    - id (question ID; optional)
    - message (question response body; required)
    - public (0 = friends only, 1 = public, 2 = question asker only; optional)

collaborate/vote

Vote a question response up or down.

- Type: POST

- Authentication: true
- Formats: JSON, XML
- Parameters:
    - id (response ID; required)
    - dir (up or down; required)

News & Events Endpoints

news/list
Retrieves a list of news/events that match the given criteria.

- Type: GET
- Authentication: false
- Formats: JSON, XML
- Parameters:
    - type (news or event; optional)
    - slug (category; optional, but must specify type)

news/show
Retrieves the specified news or event by ID.

- Type: GET
- Authentication: false
- Formats: JSON, XML
- Parameters:
    - id (news/event ID; required)
    - short_id (news/event short ID; required if id isn't given)

news/create
Creates a new news/event post.

- Type: POST
- Authentication: true
- Formats: JSON, XML
- Parameters:
    - title (required)
    - content (the link or text content; required)
    - category (ID of the category; required)
    - type (text or link; required)
    - event_location (events only, optional)
    - event_location_area (events only, optional)
    - event_all_day (1 or 0; events only, optional)
    - event_start_from_date (events only, optional)
    - event_start_from_hour (events only, optional)
    - event_start_from_minute (events only, optional)
    - event_start_from_ampm (am or pm; events only, optional)
    - event_start_to_date (events only, optional)
    - event_start_to_hour (events only, optional)
    - event_start_to_minute (events only, optional)
    - event_start_to_ampm (am or pm; events only, optional)

news/vote
Vote for the given news/event post.

- Type: POST
- Authentication: true
- Formats: JSON, XML
- Parameters:

    - id (ID of post; required)
    - dir (up or down; required)

news/create_comment

Posts a comment for the given news story

- Type: POST
- Authentication: true
- Formats: JSON, XML
- Parameters:
    - news_id (ID of post; required)
    - short_news_id (short ID of post; required if news_id isn't given)
    - reply_to (ID of comment this new one is replying to; optional if top-level comment)
    - content (comment content; required)

news/location_search

Autocomplete endpoint for location searching.

- Type: GET
- Authentication: false
- Formats: JSON
- Parameters:
    - q (search query; required)

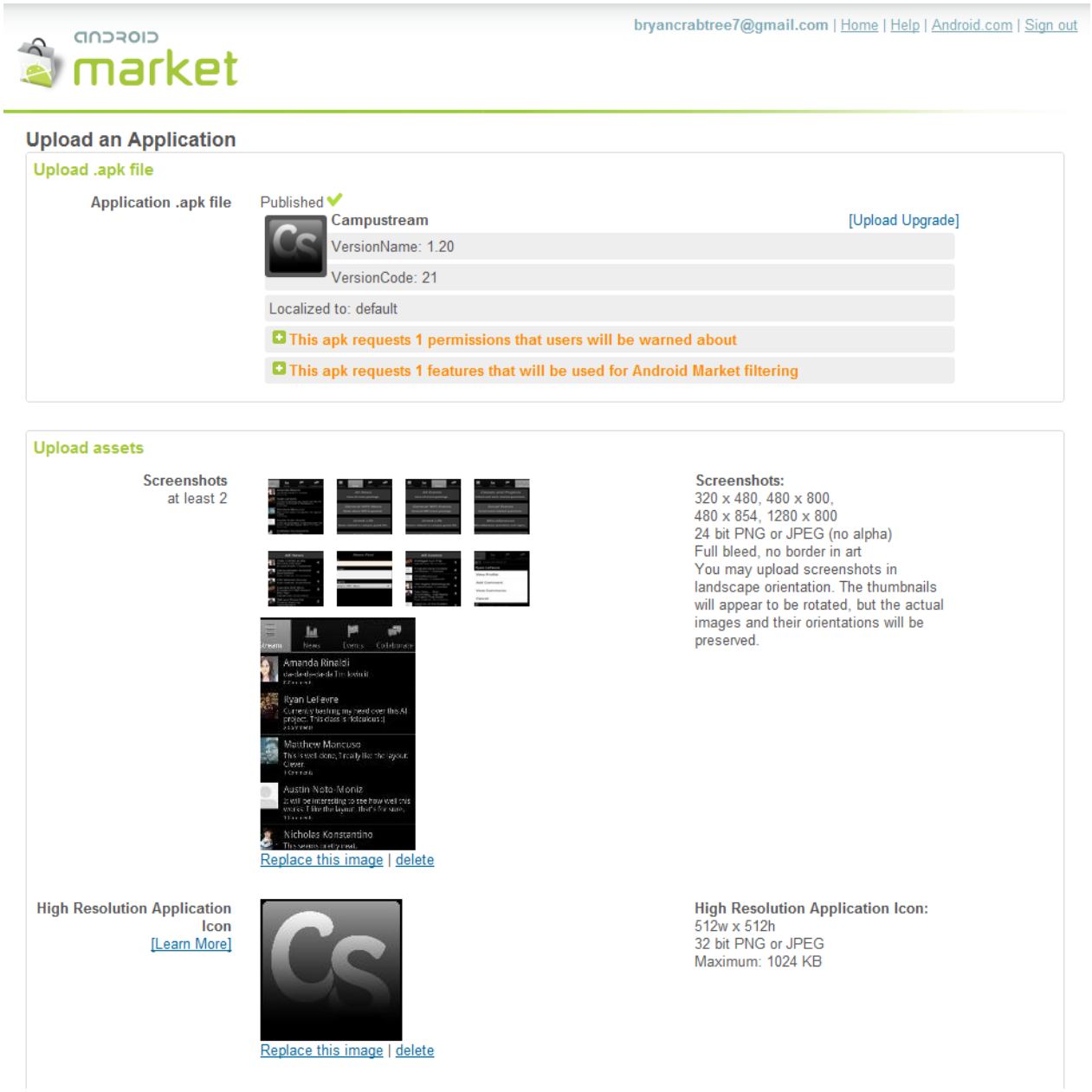
Notification Endpoints

notifications/show

Retrieves all current notifications for the logged in user.

- Type: POST
- Authentication: true
- Formats: JSON
- Parameters: None

Appendix D – Android Market Submission Process



High Resolution Application Icon
[Learn More]

Replace this image | delete

High Resolution Application Icon:
512w x 512h
32 bit PNG or JPEG
Maximum: 1024 KB

Promotional Graphic
optional

Add a promotional graphic:
Choose File No file chosen
Upload

Promo Graphic:
180w x 120h
24 bit PNG or JPEG (no alpha)
Full bleed, no border in art

Feature Graphic
optional

Add a feature graphic:
Choose File No file chosen
Upload

Feature Graphic:
1024w x 500h
24 bit PNG or JPEG (no alpha)
Will be downsized to mini or micro

Promotional Video
optional

Add a promotional video link:
http://

Promotional Video:
Enter YouTube URL

Marketing Opt-Out

☑ Do not promote my application except in Android Market and in any Google-owned online or mobile properties. I understand that any changes to this preference may take sixty days to take effect.

Listing details

Language
add language

| *English (en) |
Star sign (*) indicates the default language.

Title (en)

Campustream
11 characters (30 max)

Description (en)

```
Campustream allows you to access WPI's very own social
network. You can view and post your own news, events, and
questions for the community to see. Join your friends and
help make WPI have the best community ever.

Forget where an event is?
Have a question about something?
Need your event to get more exposure?
```

535 characters (4000 max)

Recent Changes (en)
VersionName: 1.20
[Learn More]

0 characters (500 max)

Promo Text (en)

535 characters (4000 max)

Recent Changes (en)
VersionName: 1.20
[Learn More]

0 characters (500 max)

Promo Text (en)

0 characters (80 max)

Application Type Applications

Category Social

Price Free

Publishing options

Copy Protection
- (•) Off (Application can be copied from the device)
- ( ) On (Helps prevent copying of this application from the device. Increases the amount of memory on the phone required to install the application.)

The copy protection feature will be deprecated soon, please use licensing service instead.

Content Rating
[Learn More]
- ( ) High Maturity
- ( ) Medium Maturity
- ( ) Low Maturity
- (•) Everyone

Locations **Select locations to list in:**

☑ **All locations**
(Includes more countries than those listed below. As the developer, you are responsible for complying with country-specific laws related to the distribution or sale of your application into that country, including your home country.)

Contact information

Website http://www.campustream.com

Email bryan@campustream.com

Phone

Consent

Appendix E – Pre-Development Survey

The main objective of this MQP is to test the plausibility of mobile social networks in a college environment and to see what uses are made possible by the introduction of a mobile platform. This objective is intended to increase the attendance and awareness of social events on campus as well as to provide an easier way for students to interact and stay connected with one another. That said, by answering these questions you will help us get a feel for where the main interests of WPI students lie.

* Required

Which of the following mobile devices do you own? *Select all that apply

- ☐ Android Device
- ☐ iPhone, iPad, or iPod Touch
- ☐ Blackberry
- ☐ Windows Mobile Phone
- ☐ None
- ☐ Other: _____

Do you intend to purchase any of the following mobile devices in the next six months? *Select all that apply

- ☐ Android Device
- ☐ iPhone, iPad, or iPod Touch
- ☐ Blackberry
- ☐ Windows Mobile Phone
- ☐ None
- ☐ Other: _____

Which of the following social networks do you currently participate in? *Select all that apply

- ☐ Facebook
- ☐ Twitter
- ☐ Myspace
- ☐ Foursquare
- ☐ Orkut
- ☐ Google Latitude
- ☐ None
- ☐ Other: _____

Do you actively use location-based services in social networks? *In other words, do you use services that focus on sharing information related to your current location?

- Yes
- No

On average, how frequently do you attend campus events and/or social gatherings? *Including school sponsored events, fraternity/sorority parties, club/organization events, etc.

- More than 6 times a month
- 4-5 times a month
- 2-3 times a month
- Once a month
- Less than once a month

On a scale of 1 to 5, how well connected do you feel with other students and campus events? *Do you feel like you always know what is going on around campus or do you feel like you're missing out on a lot?

	1	2	3	4	5	
Very unconnected						Very connected

On a scale of 1 to 5, how safe do you feel on campus? *Including at night time

	1	2	3	4	5	
Very unsafe						Very safe

Would you be interested in using a mobile application that aims to improve social connections on campus and provide a central place to learn about events? *

- Yes
- No

Would you be interested in a system that allows you to collaboratively ask questions among your peers? *This includes class-related questions, social questions, and anonymous questions.

- Yes
- No

Would you like to see a social application that helps report and monitor suspicious activity on campus? *In other words, an application that aims to help improve campus security collaboratively.

- Yes
- No

Would you be willing to participate in testing a social networking application on campus? *If yes, you will be asked to provide your email address.

- Yes
- No

Appendix F – Android Code Sample

The following is code from a single class from the Android application. This class generalizes the overall design of the application and gives an overview of how each activity within the application is instantiated and how the content is populated. All of the Android code for this project can be found in the "Code" folder submitted along with this MQP.

```
package com.main.campustream;

import java.util.ArrayList;
import org.json.JSONArray;
import org.json.JSONObject;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ListActivity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;

/**
 * @author Bryan Crabtree
 *
 */
public class StreamTab extends ListActivity implements OnClickListener {

	private ProgressDialog m_ProgressDialog = null;
	private ArrayList<StreamStatus> m_statuses = null;
	private StatusAdapter m_adapter;
	private Runnable viewStatuses;

	int numStatuses = 0;
	String statusList = "";

	/*
	 * (non-Javadoc)
	 *
	 * @see android.app.Activity#onCreate(android.os.Bundle)
	 */
```

```
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.streamlistview);
		m_statuses = new ArrayList<StreamStatus>();
		this.m_adapter = new StatusAdapter(this,
R.layout.streamlistviewrows, m_statuses);
		setListAdapter(this.m_adapter);

		Button statusButton = (Button) findViewById(R.id.statusButton);
		statusButton.setOnClickListener(this);

		Button refreshButton = (Button) findViewById(R.id.refreshButton);
		refreshButton.setOnClickListener(newRefreshListener);

		ListView streamList = getListView();
		streamList.setTextFilterEnabled(true);
		streamList.setOnItemClickListener(streamListListener);

		viewStatuses = new Runnable() {
			@Override
			public void run() {
				getStatuses();
			}
		};
		Thread thread = new Thread(null, viewStatuses,
"MagentoBackground");
		thread.start();
		m_ProgressDialog = ProgressDialog.show(StreamTab.this, "Please
Wait", "Retrieving Public Stream", true);
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see android.app.Activity#onCreateDialog(int)
	 */
	protected Dialog onCreateDialog(int position) {
		Dialog dialog = null;
		AlertDialog.Builder builder = new AlertDialog.Builder(this);

		final CharSequence[] items = { "View Profile", "Add Comment",
"View Comments", "Cancel" };

		builder.setTitle(m_statuses.get(position).getStatusName());
		Campustream.statusID = m_statuses.get(position).getStatusID();
		builder.setItems(items, new DialogInterface.OnClickListener() {
			public void onClick(DialogInterface dialog, int item) {
				if (item == 0) {
					OpenProfile();
				} else if (item == 1) {
					NewComment();
				} else if (item == 2) {
					OpenComments();
				}
			}
		});
```

```
            AlertDialog alert = builder.create();
            alert.show();

            return dialog;
      }

      /**
       * Dismisses the progress dialog and updates the list.
       */
      private Runnable returnRes = new Runnable() {

            @Override
            public void run() {
                  if (m_statuses != null && m_statuses.size() > 0) {
                        m_adapter.notifyDataSetChanged();
                        for (int i = 0; i < m_statuses.size(); i++)
                              m_adapter.add(m_statuses.get(i));
                  }
                  m_ProgressDialog.dismiss();
                  m_adapter.notifyDataSetChanged();
            }
      };

      /**
       * Retrieves the 50 most recent public status updates.
       */
      private void getStatuses() {
            try {
                  String statusName = "";
                  String statusUser = "";
                  String statusIcon = "";
                  String statusMessage = "";
                  String statusComments = "";
                  String statusType = "";
                  String statusID = "";
                  m_statuses = new ArrayList<StreamStatus>();
                  statusList = APIs.GetStream();
                  JSONArray followers = new JSONArray(statusList);
                  numStatuses = followers.length();

                  for (int x = 0; x < numStatuses; x++) {
                        StreamStatus currentStatus = new StreamStatus();

                        JSONObject jsonStatus = followers.getJSONObject(x);
                        JSONObject jsonUser =
jsonStatus.getJSONObject("user");

                        statusName = jsonUser.getString("name");
                        statusUser = jsonUser.getString("username");
                        statusIcon = jsonUser.getString("avatar");
                        statusMessage = jsonStatus.getString("message");
                        statusComments =
jsonStatus.getString("num_comments");
                        statusType = jsonStatus.getString("type");
                        statusID = jsonStatus.getString("id");
                        statusIcon = statusIcon.replaceFirst("medium",
"small");
```

```
                        statusName = Campustream.correctString(statusName);
                        statusMessage =
Campustream.correctString(statusMessage);

                        currentStatus.setStatusName(statusName);
                        currentStatus.setStatusUser(statusUser);
                        currentStatus.setStatusIcon(statusIcon);
                        currentStatus.setStatusMessage(statusMessage);
                        currentStatus.setStatusComments(statusComments);
                        currentStatus.setStatusType(statusType);
                        currentStatus.setStatusID(statusID);

                        m_statuses.add(currentStatus);
                  }
            } catch (Exception e) {
            }
            runOnUiThread(returnRes);
      }

      /**
       * @author Bryan Crabtree
       *
       */
      public class StatusAdapter extends ArrayAdapter<StreamStatus> {

            private ArrayList<StreamStatus> items;

            public StatusAdapter(Context context, int textViewResourceId,
ArrayList<StreamStatus> items) {
                  super(context, textViewResourceId, items);
                  this.items = items;
            }

            /*
             * (non-Javadoc)
             *
             * @see android.widget.ArrayAdapter#getView(int,
android.view.View, android.view.ViewGroup)
             */
            @Override
            public View getView(int position, View convertView, ViewGroup
parent) {
                  View v = convertView;
                  if (v == null) {
                        LayoutInflater vi = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                        v = vi.inflate(R.layout.streamlistviewrows, null);
                  }
                  StreamStatus o = items.get(position);
                  if (o != null) {
                        TextView tt = (TextView)
v.findViewById(R.id.toptext);
                        TextView bt = (TextView)
v.findViewById(R.id.bottomtext);
                        TextView lt = (TextView)
v.findViewById(R.id.lowertext);
```

```
                        final ImageView pic = (ImageView)
v.findViewById(R.id.statusIcon);
                        if (tt != null) {
                              tt.setText(o.getStatusName());
                        }
                        if (bt != null) {
                              if (o.getStatusType().equals("question")) {
                                    bt.setText("Asked: " +
o.getStatusMessage());
                              } else if
(o.getStatusType().equals("newsevent")) {
                                    bt.setText("Posted: " +
o.getStatusMessage());
                              } else if
(o.getStatusType().equals("response")) {
                                    bt.setText("Replied: " +
o.getStatusMessage());
                              } else {
                                    bt.setText(o.getStatusMessage());
                              }
                        }
                        if (lt != null) {
                              if (Integer.parseInt(o.getStatusComments()) ==
1) {
                                    lt.setText(o.getStatusComments() + "
Comment");
                              } else {
                                    lt.setText(o.getStatusComments() + "
Comments");
                              }
                        }
                        if (pic != null) {
                              Drawable picture = null;
                              Boolean pictureSet = false;
                              final String icon = o.getStatusIcon();

                              for (int y = 0; y < 1000; y++) {
                                    if (Campustream.profileAvatars[y] !=
null) {
                                          if
((Campustream.profileAvatars[y].equals(icon)) &&
(Campustream.profileAvatarStatuses[y].equals("Ready"))) {
                                                picture =
Campustream.profileIcons[y];

      pic.setImageDrawable(picture);
                                                pictureSet = true;
                                                y = 10000;
                                          } else if
((Campustream.profileAvatars[y].equals(icon)) &&
(Campustream.profileAvatarStatuses[y].equals("Downloading"))) {
                                                final int x = y;
                                                pic.setImageDrawable(null);
                                                final Handler handler = new
Handler() {
                                                      @Override
```

```
                                                      public void
handleMessage(Message message) {

      pic.setImageDrawable((Drawable) message.obj);
                                                      }
                                                };

                                                final Thread thread = new
Thread() {
                                                      @Override
                                                      public void run() {
                                                            while
(Campustream.profileAvatarStatuses[x].equals("Downloading")) {
                                                                  try {

      Thread.sleep(100);
                                                                  } catch
(InterruptedException e) {

      e.printStackTrace();
                                                                  }
                                                      }
                                                      final Drawable
newPicture = Campustream.profileIcons[x];
                                                      Message message =
handler.obtainMessage(1, newPicture);

      handler.sendMessage(message);
                                                      }
                                                };
                                                thread.start();
                                                pictureSet = true;
                                                y = 10000;
                                          }
                                    } else {
                                          y = 10000;
                                    }
                              }

                              if (pictureSet == false) {
                                    for (int y = 0; y < 1000; y++) {
                                          if (Campustream.profileAvatars[y]
== null) {
                                                Campustream.profileAvatars[y]
= icon;

      Campustream.profileAvatarStatuses[y] = "Downloading";
                                                final int x = y;
                                                pic.setImageDrawable(null);
                                                final Handler handler = new
Handler() {
                                                      @Override
                                                      public void
handleMessage(Message message) {

      pic.setImageDrawable((Drawable) message.obj);
                                                      }
```

```
                                                };

                                                Thread thread = new Thread()
{
                                                      @Override
                                                      public void run() {
                                                            while
(Campustream.profileIcons[x] == null) {

      Campustream.profileIcons[x] = GetUserInfo.GetAvatarIcon(icon);
                                                            }

      Campustream.profileAvatarStatuses[x] = "Ready";
                                                            final Drawable
newPicture = Campustream.profileIcons[x];
                                                            Message message =
handler.obtainMessage(1, newPicture);

      handler.sendMessage(message);
                                                      }
                                                };
                                                thread.start();
                                                y = 10000;
                                          }
                                    }
                              }
                        }
                  }
                  return v;
            }
      }

      /**
       * Refreshes the list.
       */
      public void refreshList() {
            m_adapter.clear();
            viewStatuses = new Runnable() {
                  @Override
                  public void run() {
                        getStatuses();
                  }
            };
            Thread thread = new Thread(null, viewStatuses,
"MagentoBackground");
            thread.start();
            m_ProgressDialog = ProgressDialog.show(StreamTab.this, "Please
Wait", "Retrieving Public Stream", true);
      }

      /**
       * Opens the profile of the acted upon user.
       */
      public void OpenProfile() {
            Intent myIntent = new Intent(this, Profile.class);
            startActivityForResult(myIntent, 0);
      }
```

```
	/**
	 * Click listener for the refresh button.
	 */
	private OnClickListener newRefreshListener = new OnClickListener() {
		public void onClick(View v) {
			refreshList();
		}
	};

	/**
	 * Click listener for the list view.
	 */
	public OnItemClickListener streamListListener = new
OnItemClickListener() {
		public void onItemClick(AdapterView<?> a, View v, int position,
long id) {
			Campustream.currentProfileUsername =
m_statuses.get(position).getStatusUser();
			onCreateDialog(position);
		}
	};

	/*
	 * (non-Javadoc)
	 *
	 * @see android.view.View.OnClickListener#onClick(android.view.View)
	 */
	public void onClick(View v) {
		Intent myIntent = new Intent(this, StatusDialog.class);
		startActivityForResult(myIntent, 0);
	}

	/**
	 * Opens a dialog to comment on a status.
	 */
	public void NewComment() {
		Intent myIntent = new Intent(this, CommentDialog.class);
		startActivityForResult(myIntent, 0);
	}

	/**
	 * Opens the comment list for the acted upon item.
	 */
	public void OpenComments() {
		Intent myIntent = new Intent(this, CommentList.class);
		startActivityForResult(myIntent, 0);
	}
}
```

Appendix G – PHP Code Sample

```
<?
/**
 * application/controllers/user.php
 *
 * Handles all interactions with users on Campustream
 */
class User_Controller extends Controller implements REST {
	public $enable_session = true;
	public $template = 'template/main';

	/**
	 * Authorizes a user given a username and password. Used by the API
	 * since it returns an access token.
	 */
	public function authorize() {
		$username = $_POST['username'];
		$password = $_POST['password']; // Must be sha1 hashed already

		if (!$username || !$password) {
			return Hub::http_error(401, "Bad request");
		}

		try {
			$user = User_Model::find_user_for_login($username, $password, true);

			$access_token = Session_Controller::generate_access_token($user);

			View::respond_to('html', function () use($access_token) {
				echo $access_token;
			});

			View::respond_to('json', function () use($access_token) {
				echo json_encode(array('access_token' => $access_token));
			});

		} catch (Exception $e) {
			return Hub::http_error(403, "Invalid credentials");
		}
	}

	/**
	 * Shows the given user's profile page.
	 */
	public function show($args) {
		if (isset($args['username'])) {
			$username = $args['username'];
		} elseif (isset($_GET['username'])) {
			$username = $_GET['username'];
```

```
	} elseif (isset($_POST['username'])) {
		$username = $_POST['username'];
	} else {
		View::respond_to('html', function () {
			echo "Not found";
		});

		View::respond_to(array('json','jsonp','xml'), function () {
			Hub::http_error(401, "Missing username");
		});

		return false;
	}

	$user = ActiveCache::find('User_Model', "name:$username", 43200)->sql(
		"SELECT * FROM users WHERE username = '$username' LIMIT 1"
	);

	if(!$user->is_loaded()) {
		View::respond_to('html', function() {
			echo "User not found.";
		});
		View::respond_to(array('xml','json', 'jsonp'), function() {
			Hub::http_error(404, "User not found");
		});

		return false;
	}

	$template = $this->template;
	View::respond_to('html', function() use($user, $template) {
		$view = new View('user/show');
		$view->user = $user;

		if (sess::isActiveUser($user)) {
			$template->active = 'profile';
		}

		$template->content = $view->render();
		echo $template->render();
	});

	View::respond_to(array('xml','json','jsonp'), function($format) use($user) {
		echo $user->{"to_$format"}();
	});
}

/**
 * Pretty API endpoint for redirecting to a user's avatar. Used for the API.
 */
```

```
public function avatar($args) {
	$size = $args['size'];
	$username = $args['username'];

	$user = ActiveCache::find('User_Model', "name:$username", 43200)->sql(
		"SELECT * FROM users WHERE username = '$username' LIMIT 1"
	);

	if (!$user->is_loaded()) {
		return Hub::http_error(404, "User not found");
	}

	Hub::redirect($user->avatar_url($size));
}

/**
 * Lists users of the site.
 */
public function list_users() {
	sess::require_login();

	$view = new View('user/list');
	$this->template->title = "Users";
	$this->template->active = 'users';
	$this->template->content = $view->render();
	echo $this->template->render();
}

/**
 * Loads all users whose names start with the given letter.
 */
public function show_list() {
	if (!$this->session->get('authenticated')) {
		return Hub::http_error(403, "Unauthorized");
	}

	if (!isset($_POST['letter']) || strlen($_POST['letter']) != 1) {
		return Hub::http_error(401, "Missing or invalid data");
	}

	$letter = strtolower($_POST['letter']);

	$r = RedisManager::connection();
	$user_ids = $r->smembers("users:by_first_letter:$letter");

	$users = array();
	foreach ($user_ids as $id) {
		$user = ActiveCache::find('User_Model', $id, 43200)->sql(
			"SELECT * FROM users WHERE id = $id LIMIT 1"
		);
```

```
                if ($user->is_loaded()) {
                    $users[] = $user->limited_object();
                }
            }

            if (count($users) > 0) {
                util::objectSort($users, 'name');
            }

            View::respond_to(array('json', 'xml'), function ($format) use($users) {
                if ($format == 'xml') {
                    echo xml::encode_array($users, 'users', 'user');
                } else {
                    echo json_encode($users);
                }
            });
        }

}
```